

تحليل وتصميم الخوارزميات

Algorithms Design and Analysis

تأليف

الدكتور	الأستاذ المساعد	الباحث
حسن ياسين طعنه	هند رستم محمد شعبان	حسن ثابت رشيد كرماشة

hind_restem@yahoo.com
hassan_thabit@yahoo.com

تحليل وتصميم الخوارزميات
Algorithms Design and Analysis

الفهرس

الفصل الأول: مقدمة (Introduction)

- 1-1: مقدمة في الخوارزميات (Algorithms Introduction)
- 2-1: كيفية تحليل الخوارزمية (Algorithm Analysis)
- 3-1: الوقت الكلي لتنفيذ الخوارزمية (Execution Time)
- 4-1: الحالات الأفضل والأسوأ والمتوسطة للتحليل (Best & Worst & Average Cases Analysis)
- 5-1: الصيغ التقريبية (Asymptotic notation)
- 6-1: الصيغ الشائعة لأوقات التنفيذ (The Times of Executive Notation)
- 7-1: الاستدعاء الذاتي لشجرة التسيطات أو الاستدعاءات (Recursion) (Tree)
- 8-1: قياس الانجازية (Performance Measurement)

الفصل الثاني: الترتيب (Sorting)

- 1-2: خوارزميات الترتيب (Sorting Algorithms)
- 2-2: أنواع الترتيب (Types of Sorting)
- 3-2: خوارزميات الترتيب الداخلي (Internal Sort Algorithms)
 - 1- ترتيب الاختيار (Selection Sort)
 - 2- ترتيب الفقاعي (Bubble Sort)
 - 3- ترتيب الإضافة (Insertion Sort)
 - 4- ترتيب شيل (Shell Sort)
 - 5- الترتيب السريع (Quick Sort)
 - 6- ترتيب الأساس (Radix Sort)
 - 7- ترتيب المؤشرات (Pointers Sort)
 - 8- ترتيب الشجري لشجرة البحث الثنائية (Tree Sort)
 - 9- Topological sorting
- 4-2: خوارزميات الترتيب الخارجي (External Sort Algorithms)
 - 1- ترتيب الدمج (merge Sort)
 - 2- ترتيب الدمج المتوازن ذو الممرتين (Balanced Two-Way Merge Sort)

الفصل الثالث: البحث (Searching)

- 1-3: البحث (Searching)
- 2-3: البحث التسلسلي (Sequential Search)
- 3-3: البحث الثنائي (Binary Search Algorithm)
- 4-3: البحث في الشجرة الثنائية (Binary Search tree)
- 5-3: تعقيد خوارزمية البحث (Algorithm search Complexity)

الفصل الرابع: الأمثلية في مسائل تصميم الخوارزميات (Optimization in Algorithms Design Equations)

- 1-4: المخططات (Graphs)
- 2-4: أنواع المخططات (Type of Graphs)
- 3-4: طول المسار (Path Length)
- 4-4: طريقة الجشع أو الطماع (Greedy Method)
- 5-4: مسألة الجراب (Knapsack Problem)
- 6-4: استخدام قاعدة الطماع في إيجاد أمثلية البيانات

الفصل الخامس: البرمجة الديناميكية (Dynamic Programming)

- 1-5: البرمجة الديناميكية (Dynamic programming)
- 2-5: أمثلة على البرمجة الديناميكية
- 3-5: تجميع البيانات (Data clustering)
- 4-5: خوارزمية (Dijkstra)
- 5-5: أمثلة لتطبيق خوارزمية (Dijkstra)
- 6-5: المخططات المتعددة المراحل (Multistage graph)
- 1-6-5: الطريقة التصاعدية (Forward approach)
- 2-6-5: الطريقة التناقصية (Backward approach)
- 3-6-5: طريقة اقتفاء الأثر رجوعاً (Back Tracking)

الفصل الأول

مقدمة

(Introduction)

1-1: مقدمة في الخوارزميات (Algorithms Introduction) :

- الخوارزمية هي مجموعة محددة من التعليمات (خطوات الحل) التي تؤدي إلى إنجاز وظيفة (مهمة) معينة ويجب أن تتوافق فيها الشروط التالية :
1. المدخلات (Input) : صفر أو أكثر من القيم .
 2. المخرجات (Output) : قيمة واحدة على الأقل .
 3. الوضوح (Definiteness) : كل خطوة فيها (الخوارزمية) واضحة المعاني وغير غامضة أي يجب أن نفهم من قبل جميع الناس (مفهوم الحسابات).
 - و على سبيل المثال نأخذ العبارة "Add 6 or 7 to X" هذه العبارة غير مفهومة لأنها لا تحدد الخوارزمية لأنها عبارة غير واضحة .
 4. المحدودية (Finiteness) : كل خطوات الخوارزمية يمكن حلها في فترة زمنية محددة ، والتوضيح تلك نأخذ العبارة " قسم الرقم (10) على (3) بدقة عالية (كاملة) " هذه العبارة غير محدودة ويجب أن لا يسمح بها دخال البرنامج .
 5. الفعالية (Effectiveness) : كل خطوة تكون ممكنة الحل أو التنفيذية ، مثال تلك العبارة " $0/3$ " لا يمكن حلها أبداً .

يمكن لنا أن نوضح الفرق بين الخوارزمية والبرنامج حيث أنه في النظرية الاحتمالية يوجد فرق بين الخوارزمية والبرنامج ، ففي الخوارزمية يجب أن تتوافق الشروط الخمسة الأتفة للذكر ويمكن وصفها بطرق عديدة مثل لغة طبيعية مع التأكيد على شرط الوضوح ، لغة خوارزمية (Pseudo-code) ، مخططات انسيابية (Flow chart) ، بينما يمكن في البرنامج عدم تحقق الشرط الرابع حيث أن نظام التشغيل هنا هو الذي يعتمد على البرنامج ويوصف البرنامج بلغة الحاسبة حيث أنه يصمم ليتحكم في تنفيذ مجموعة من الأعمال (Jobs) بحيث عند عدم توفر عمل معين فإنه لا ينتهي من أعماله بل يستقر ويدخل في حالة لتتظار لحين إدخال عمل جديد .

إن لكل لغة برمجية يوجد مترجم أو مفسر ولا يمكن تواجدهما معاً حيث إن المفسر يقوم بتنفيذ البرنامج خطوة خطوة (step by step) بينما المترجم فإنه يتخذ البرنامج كاملاً ويظهر النتائج والأخطاء ، هذا يعني أن البرنامج هو عبارة عن خوارزمية وهيكل يأتى أي أنه طريقة لتنظيم البيانات.

خطوات تطوير البرنامج :

تتم عملية تطوير البرنامج بخمس خطوات رئيسية هي :

1. توصيف المتطلبات (Requirement specification):

هو تحديد المدخلات والمخرجات.

2. التصميم (Design):

هو تحديد العمليات الرئيسية التي تطبق على كل بيان بياني والفقرات وجرد أجهزة معالجة لتنفيذ هذه العمليات.

3. التحليل (Analysis):

هو المفصلة بين الخوارزميات المعروفة التي تحل نفس المسألة تبعاً لمقاييس المفصلة متقناً عليها (تقديرات الوقت وتعقيدات الفراغ (الخرن)) باختيار أفضلها .

4. التحسين والتشفير (Refinement & Coding):

في هذه الخطوة يتم تحديد التمثيل الباقي لكل بيان ثم كتابة الإجراءات لكل عملية على تلك الكيفيات وتكون نسخة متكاملة للبرنامج.

ملاحظة // التحليل يصلح الأخطاء اعتماداً على تعقيدات الخرن والوقت بينما التحسين يصلح الأخطاء اعتماداً على النتائج الظاهرة في نهاية البرنامج.

5. التحقق من الصلابة (Verification):

تضمن هذه الخطوة ثلاث جوانب هي :

أ- البرهنة على الصحة (Proving) :

قبل استخدام البرنامج يجب إثبات أنه صحيح حيث يتم استخدام الطرق المعروفة للبرهنة على الصحة.

ب- الاختبار (Testing):

هي عملية توليد نماذج بيانية يعمل عليها البرنامج حيث إن الهدف منها هو إعطاء إشارة على وجود أخطاء في البرنامج.

ج- تشخيص الأخطاء (Debugging):

عملية تحديد مواقع الأخطاء البرمجية في البرنامج وتصحيحها .

ملاحظة : إن التعريف يختلف عن الصلابة فالترقيق هو معرفة شيء قد يكون صحيح أو خطأ بينما الصلابة هي معرفة شيء يجب أن يكون صحيح .

أما النموذج فهو تحقيق تمثيل بياني بالشكل الصحيح.

في علم الحاسبات يتم أولاً الاختبار وبعدها يتم البرهنة بينما في علم الرياضيات يتم البرهان وبعده الاختبار.

2-1: كيفية تحليل الخوارزمية (Algorithm Analysis)

تحليل الخوارزمية هو تحديد الكفاءة للخوارزمية ومن ثم تصنيفها حيث يوجد مقياسين مرتبطين مباشرة بشجرة الخوارزمية هما :

1 - مقياس تعقيدات الفراغ أو التخزين (Space Complexity): هي كمية الذاكرة التي يتطلبها تشغيل البرنامج حتى اكتماله بحيث يعتمد هذا النوع على جزئين :

أ- جزء ثابت : هو مستقل عن خصائص المدخلات والمخرجات حيث يتضمن هذا الجزء فراغ التعليمات (Code space) ، الفراغ المخصص للمتغيرات (Data Space) سواء كانت البسيطة أو المتغيرات المركبة ذات الحجم الثابت إضافة إلى فراغ التراكيب ، الخ .
 ب- جزء متغير: يتألف من الفراغ الذي يتطلبه البرنامج بالمتغيرات المركبة والتي يعتمد حجمها على مثال المسألة المراد حلها ، إضافة إلى فراغ المكس المستخدم في التداخل (Reaction).



شكل (1): تحليل الخوارزمية

إن التخزين التبادلي يمكن توضيحه بالمتغيرات التي يدخلها البرنامج (أي يدخل قيعها) وكذلك يتحكم بأسمائها فهي متعدة على إدخال البرنامج.

أما القيم المركبة فهي المصفوفة التي تمثل بالمكس حيث المؤشر هو (Sip) عمادياً وبرمجياً يسمى (Top) ، وفيما يخص تصنيف الخوارزمية فإن المهم هنا هو مستوى المصفوفة أي المكس.

يمكن صياغة تعقيدات الخزن للبرنامج كالآتي :

ثابت

Code segment
Data segment
Heap segment
Stack segment

إن جزء (Code Segment) يمكن تمثيله كما الحوال الجاهزة ، أما (Heap Segment) فيكون المتغيرات التي يستخدمها البرنامج .

وعليه يمكن صياغة تعقيدات الفراغ $S(p)$ للبرنامج (p)

$$S(p) = \text{Const} + Sp$$

حيث إن :

Const : تمثل جزء (Code segment) والمتغيرات البسيطة .

Sp : تمثل خصائص المثال .

2- تعقيدات الوقت (Time Complexity) : هي كمية الوقت التي يتطلبها تنفيذ البرنامج حتى اكتماله ويتألف من :

$$T(p) = \text{Const} + tp$$

حيث :

Const : يمثل ثابت خاص بوقت الترجمة أو التفسير .

Tp : يمثل وقت تشغيل البرنامج .

مثال 1 // لبيان تعقيدات الفراغ (الخزن) والوقت لدالة معينة (بلغة C++) :

```
Float abc(float a,float b,float c)
{return(a+b+5*c+(a+b+c)/(a+b)+4.0); }
```

تعقيدات الفراغ أو الخزن :

تطلب الدالة (abc) خمسة خلايا خزن لثمة قيم المتغيرات (a,b,c) والمتغير الذي يحمل اسم الدالة وعنوان العونة (Return address) وهو خزن ثابت لا يعتمد على خصائص المثال (a,b,c).

$$S_{abc}(a,b,c) = 0$$

إن قيمة الصفر هنا تعني إن الخزن ثابت أي لا يعتمد على خصائص المثال .

تعقيدات الوقت : تستخدم صيغة عد الخطوات (Steps Count) لقياس تقدير الوقت حيث إن عدد الخطوات لهذه الدالة يساوي واحد ، ولها فإن :

$$T_{abc}(a,b,c) = 0$$

إن قيمة الصفر هنا أيضا تعني إن الوقت ثابت .

مثال //2 اكتب خوارزمية لإيجاد القاسم المشترك الأعظم (Greatest Common Divisor) لعددين صحيحين .

//الحل

```

Step 0 : [ check m and n ]
    If m <= 0 or n <= 0 then
        Print error.
Step 1: [ test m and n ]
    If m < n then
        Inter change m by n.
Step 2: [find the remainder]
    Divid m by n and let r is
    Remainder we will have 0 <= r < n.
Step 3: [is r = zero]
    If r = 0 then
        GCD = n and exit.
Step 4:[ inter change ]
    M ← n, n ← r go to step 2
    
```

مثال تطبيق 1//

m	n	r
10	6	4
6	4	2
4	2	0

GCD = 2

مثال تطبيق 2 //

m	n	r
20	130	
130	20	10
20	10	0

GCD = 10

عدد مرات تنفيذ العبارة (Frequency Count) :

إن عدد مرات تنفيذ العبارة يختلف حسب عينة البيانات . حيث يوجد لدينا ما يسمى بوقت التنفيذ المفرد للعبارة (execution time for single).

Total execution time = frequency count * execution time for single

إن الوقت الكلي لتنفيذ يعتمد على العوامل التالية :

1. نوع الحاسبة (Computer type).
2. لغة البرمجة (Programming language).
3. الوقت التنفيذي الخاص لكل عبارة (Total execution time).
4. نوع المترجم أو المفسر (Compiler and interpreter).

مثال 3// افترض وجود الأجزاء البرمجية الآتية مرقبة من 1 إلى 3 كالآتي :

مثال 1	$x = x++;$	$F_c = 1$
مثال 2	$\text{For}(\text{int } i=1; i \leq n; i++)$ $x = x++;$	$F_c = n$
مثال 3	$\text{For}(\text{int } i=1; i \leq n; i++)$ $\text{For}(\text{int } j=1; j \leq n; j++)$ $x = x++;$	$F_c = n^2$

لو افترضنا أن ($n = 10$) فإن مثال رقم (2) فيه عدد مرات تكرار الخطوة التنفيذية هو (10) وعدد المرات في المثال رقم (3) هو (100).
نستنتج من ذلك أن المثال رقم (1) ينفذ أسرع من المثالين (2) و (3) ومثال (2) أسرع من مثال (3).

3-1: الوقت الكلي لتنفيذ الخوارزمية (Execution Time):

تنظيم الترتيب للخوارزمية (Order of magnitude of Algorithm) :
هو مجموع تكرارات جميع العبارات التنفيذية التي يسويها يحدد التقدير المسبق لوقت تنفيذ الخوارزمية.
إن العبارة الغير تنفيذية تعني العبارة التي ليس لها تأثير على البرنامج مثل عبارة التعليق في أي لغة برمجية يمكن استخدامها.

مثال// لدينا مصفوفة A_{ij} ، أحسب مجموع كل صف واخزن قيمته في مصفوفة اسمها S ، ثم احسب المجموع الكلي لعناصر المصفوفة A ، ثم اعطي عدد تكرارات المرات .

$$\text{Sum} = \sum_{j=1}^n a_{ij}$$

الحل // توجد طريقتين:

```

1 Grandtotal = 0;
For(int k = 1; k <= n; k++)
{ s[k] = 0;
For(int j = 1; j <= n; j++)
{ s[k] = s[k] + a[k,j];
Grand total = (Grand total + s[k]);
}
}

```

نلاحظ ان عدد التكرارات يساوي $2n * n$

```

2- Grandtotal = 0;
For(int k = 1; k <= n; k++)
{ s[k] = 0;
For(int j = 1; j <= n; j++)
{ s[k] = s[k] + a[k,j]; }
Grand total = Grand total + s[k];
}

```

وهنا نلاحظ انها تساوي $n^2 + n$

مثال ٢: ما كلفة خوارزمية CN وخوارزمية ثنائية CN^2 عموماً إن C هي ثابت ، قم بعمل مقارنة بين الخوارزميتين من حيث وقت التنفيذ عموماً إن:

C الأولى = 10 ، والثانية = 0.5

و $n = \{ 1, 5, 10, 15, 20, 25, 30 \}$

n	CN	CN ²
1	10	0.5
5	50	12.5
10	100	50
15	150	112.5
20	200	200
25	250	312.5
30	300	450

ملحظة // إذا كانت قيمة n أقل أو تساوي 20 فإن وقت الخوارزمية الثانية أقل من وقت الخوارزمية الأولى أما عند قيمات التكرار أو التعيينات أقل ، يمكن بعد هذه القيمة (20) أي (25, 30) فإن وقت الخوارزمية الأولى يكون أقل ، هذه المرة (يسج العكس)

هدف اليوم بالحصة وقت التعداد للخوارزمية خطي هناك إننا نجد $(O(g(n)))$ وهذا يعني إن وقت تنفيذ دنا يستغرق أكثر من $(C * g(n))$ حيث إن (C) هو كمية ثابتة وإلّا n تعقل عدد التعيينات المطلوبة لمحوط الخوارزمية لتختلف حصص m .

مثلاً:

- 1 $O(1)$ معنى ذلك أن وقت الحساب ثابت مثل $(x \rightarrow x+1)$ صغر البرنامج
- 2 $O(n)$ وهي أن وقت الاحتساب ذو طبيعة خطية مثل طباعة جميع عناصر مصفوفة حجمها n أو مثلاً إيجاد عنصر في قائمة موصولة
- 3 Quadratic (وقت تربيع $O(n^2)$) مثل وقت ترتيب عناصر قائمة باستخدام عناصر قسمة صفائي .
- 4 $O(n^3)$ في الوقت الترتيب لحمل عناصر مصفوفة $(n \times n \times n = 0)$
- 5 $O(2^n)$ يعني أن الوقت الترتيب لحمل جميع عناصر مصفوفة مثلاً مساريًا للعنصر هو استخدام الصيغة الأسية .
- 6 $O(\log(n))$ في الصيغة مثل وقت البحث باستخدام صيغة التوابع ثنائية مثل تلك للوصول إلى عقدة معينة في شجرة ثنائية.

ملاحظة: قيمة \log دائماً تكون مقصورة بين (0) و (1) أي أقسام عشرية .

مثال // هناك اقيم لتاليه $n = \{1, 2, 4, 8, 16\}$
 مقوم بتضمينها على نحو زمني الآليه وفازتها بجداول تعرض توضيحها .

n	$\log_2 n$	$N \log_2 n$	N^2	N^3	2^n
1	0	0	1	1	2
2	1	2	4	8	4
4	2	8	16	64	16
8	3	24	64	512	256
16	4	64	256	4096	65536

$$\log_2(x) \quad \log_{10}(x) \times 3.322$$

مربعين // الزمن التوال الساعه ووضح عصفه الفرق بين الخوارزميات

- ملاحظة 1/ وقت $O(n)$ ووقت $O(n \log n)$ يزيد كل منهما بصورة أبطأ من التوال الأخرى .
- ملاحظة 2/ عندما يكون حجم تسلسل كثير يصبح الخوارزمية لها بعد وقت كثير مثلاً الخوارزمية التي عدد عناصرها $O(n \log n)$ تكون على القيمة أو غير عصفه .
- ملاحظة 3/ لمر زمنية التي وقت تنفيذ بالصيغة الأسية فإنه يمكن اعتمادها فقط إذا كانت قيمة (n) صغيرة، حيث أن كانت قيمة n صغيرة فإن عدد الحساب قبل والتالي في وقت التعداد أسرع والتالي الغير ممكن واضح .

مثال // هناك لساله الآليه

$$F = a + b^2 + c + (a + b) \cdot (a + b) + 4 \cdot 0$$

عفاً إلى التوال $(a=2.0, b=3.0, c=0.0)$.
 المطلوب / تحديد تعقيدات الوقت وتحديد الحرر عفاً إلى (a, b, c) في قيم حقيقيه

ملاحظة: عند التعويض بالمتغير $\frac{2}{3} = 2.33$ ، الخزن يعطي 2.33 في المتغيرات الموجودة في

السؤال *

مصفوفات الحرج - يوجد لدينا مصفوفة حاليًا حرجية هي a, b, c وعكسها العكس F وتعتبر الاسم للصفة

، معنى ذلك إن $(c, b, a) = S$ أي لا يوجد جزء من S هو أي المتغير لك

في حالة إن تكون المصفوفات هذه الدالة n من العنصر ما هي مصفوفات الخزن هنا وعملها في

n=3

و (a, b, c) قيمها كالتالي.

a	b	c
3	2	1
2	3	2
1	4	1

للحل نقوم بالتالي -

نقوم بعمل جدول كالتالي - معنى ذلك أنه يوجد قيم صغيرة بالمصفوفات فالنتيجة لا يساوي صفر
ونعتمد على قيم المصفوفات

مصفوفات قوف

في الحالة الأولى وفي الدالة يساوي واحد (Count=1) لأنها تتغير مرة واحدة وهي الحالة الثانية
التي تعتمد على عدد المصفوفات (Count=n)

مثال: إيجاد مجموع عناصر مصفوفة بحدة الحد كفي في مصفوفة الدالة

$$Sum = \sum_{i=1}^n a_i$$

Float Sum(float a[], int n)

{ float S=0.0 // (عدد المصفوفات)

For(int i=1; i<=n; i++) // (عدد المصفوفات: n)

S+=a[i]; // (n)

Return S // (1)

}

يقسم مثال المسألة بالمتغير (n)

مصفوفات الحرج - نطلب الدالة (Sum) متعة حاليًا حرجية لحرج قيم (I, S, n) وعنوان المصفوفة

[a والسفر التي تجعل اسم الدالة المتكامل في عنوان الدالة]

وهو خزن ثابت لا يعتمد على حصة المتغير (أي لا يعتمد على تغير قيمه n)

$$S_{sum}(n)=0$$

تعريف الوقت

$$T_{sum}(n)=2n+3$$

لاحظ إن العلاقة التي تربط الوقت بعدد العناصر هي علاقة خطية وهي أفضل من التربيعية
مثلاً، سوف نجد نفس العدد السابق ولكن هذه المرة بطريقة الاستدعاء الذاتي (Recursion).

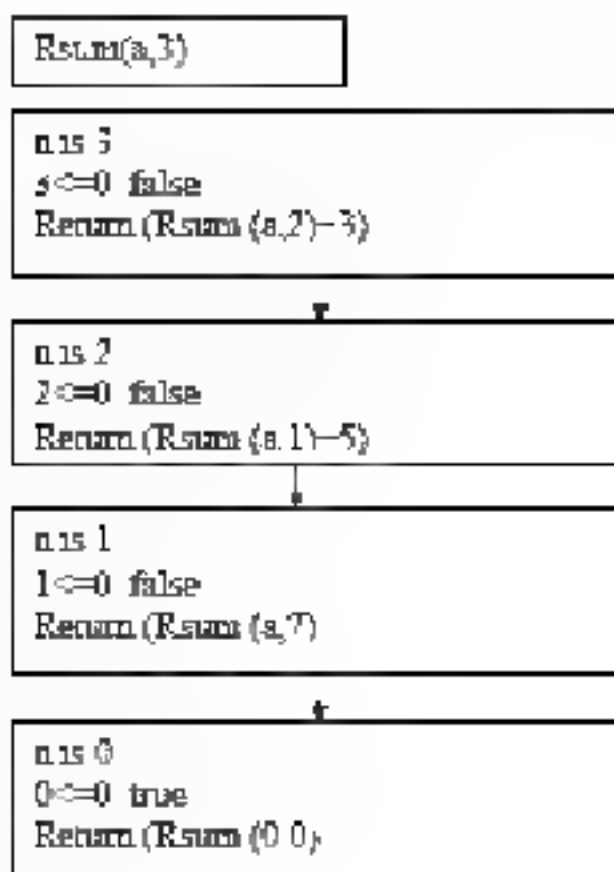
$$Sum = \sum_{i=1}^n a_i$$

$$Rsum(a, n) = \begin{cases} 0.0 & \text{if } n \leq 0 \\ Rsum(a, n-1) + a[n] & \text{if } (n > 0) \end{cases}$$

إذا بالأسطة الصغيرة نلاحظ أنها بالأسطة الصغيرة للفرمجة هي .

```
Float Rsum(float a[] ,int n)
{if (n<=0) return (0.0);
 Else return (Rsum(a,n-1)+a[n]);
}
```

ونفترض إن المتغيره (a[1..3]=3 5 7) هي إلى (n=3)



بمفاتيح الفرع

نضمن فراغ مكان الدخول المتعاقبات الشكلية والمخبرات العددية وندوي للبريد
كل تنسيق (استدعاء) يتطلب أربع حالات حرجية (حيزه بقيه (D) وحيزه للفرع إلى العصفرة
(E) وحيزه للتعديل الذي يحفل اسم ندائه بالإصافة إلى حيزه بطول العود
وهو في معنى السجل (عدد الاستدعاءات) نصب كالتالي.

معوا السجل (الاستدعاء) = الحجم الإجمالي في بول السجل - الحجم النهائي في آخر سجل
1 + |

عق للتلط (الاستدعاء) = n + 1 + 4

حدث في الحجم الأول في أول سجل بعد به عدد العصور n ثم نأخذ السجل في الحجم
النهائي في آخر تنسيق للمعك إن يكون (D) أو في هذه أخرى.

$$T_{max}(n) = 4(n-1)$$

من المعقولة أن نلاحظ في الختي ندوة خطية بعد على قعة (n) حدث إذا ارجاب قعة
(n) أن لا الحرج أن إن قة قعة لا (n) كل للحرج

بعض الأوقات مرفوع قوم باستدعاء علاقة السجل (Recurrence relations) بحسبها
كالتالي

$$f_{max}(n) = \begin{cases} 2 & \text{if } n \leq 0 \\ 2 + f_{max}(n-1) & \text{if } n > 0 \end{cases}$$

إن نقيع (2) نأكل أربعين لفظلوب لاستدعاء كل تنفيذ ، لف (f_{max}(n-1)) فهي نأكل وقت
الذي يكون قص استدعاء
إن خطوة (Else) بعد خطوة معالجة هناك فهي نأخذ القعة (D) وهي الخطوة التي
تحويلاً نأخذ نقيع (I)
ولكن هذه العلاقة الداخلية تستخدم طريقة من طرق الحل وهي طريقة بدروس التكراري
(Iterative Substitution) كالتالي

$$\begin{aligned} f_{max}(n) &= 2 + f_{max}(n-1) \\ &= 2 + 2 + f_{max}(n-2) \\ &= 2(2) + f_{max}(n-2) \\ &= 2(2) + 2 + f_{max}(n-3) \\ &= 3(2) + f_{max}(n-3) \\ &= m(2) + f_{max}(n-m) \end{aligned}$$

و عندما $(m=n)$ تأتي

$$\begin{aligned} & 2n + t_{\text{main}}(n + n) \quad , \quad n > 0 \\ & = 2n + t_{\text{main}}(0) \\ & = 2n + 2 \end{aligned}$$

بعض المتغيرات في تكون ثابت $(n=n)$ قد تكون ثابتة $(n=1)$ و أي قيمة تحتوي بحيث في
القيم في تكون ثابت $(n=n)$.

مثلاً : إذا كان مجموع عنصرين متتاليين كما في المتسلسلة n فية :

$$C_{m+n} = A_{m+n} + B_{m+n}$$

```
Void Add(type a[][size], type b[][size], type c[ ][size] in m,mi,n)
{ for(int i=1;i<=mi;i++) ... mi+1
  For (int j=1;j<=ni;j++) ... Sn
    C[i,j]=a[i,j]+b[i,j]
}
```

نلاحظ في هذا المثال يتصلب بالمتغيرات (m,n) حيث :

بعض المتغيرات : مثلاً مثلاً (m,n) نفس ذلك حزمة غير قيم المتغيرات (m,n,b,a)
بالإضافة إلى عنوان الذاكرة ونلاحظ في الخوار لا يتوقف على خصائص المتغير في إلى

$$S_{\text{main}}(m, n) = 0$$

في العادة إلى بعض الوقت في كالتالي

$$m + 1$$

$$5m$$

$$(m + 1 + m) m$$

$$(2m + 1) m$$

$$2mm + m + m + 1$$

$$T_{\text{main}}(m, m) = 2mm + 2m$$

في هذه العلاقة تكون متعلقة في حصة $(m=n)$ أما عندما تكون $(m=n)$ فإنه يتصلب خلال
تسمى لا (For) في ذلك حفظ تعقيد الذاكرة لتصبح :

$$T_{\text{main}}(m, m) = 2mm + 2m + 1$$

هناك أربع تعقيدات الزمن والوقت سلسلة عداد فيبوناتشي (Fibonacci) التي هي متتالية من الأعداد ذات كسري.
أول عنصر قيع هو (0,1) وهما يندرج في المتتالية حيث أن عدله أحصاه يتم على التتابع
التيهيه من خلال جمع العنصر السابقين كالتالي
0, 1, 1, 2, 3, 5, 8, 13.

جد "n" كل حد عند سبر الحصول عليه من خلال جمع العنصر السابقين به هذا كل (Fn)
يعمل الحد الأول في المتتالية قيع ويصورة عامة

$$F_0 = 0$$

$$F_1 = 1$$

$$F_n = F_{n-1} + F_{n-2}, n \geq 2$$

إن طريقة عمل أو تنفيذ البرنامج يتم بهندال عدد صحيح موجب ويتك (n) ويخرج قيمة (Fn)
له حيث إن كانت (n=3) فإن (Fn=3) أو (n=4) فإن (Fn=4) أو (n=5) فإن (Fn=5)
إن جزء البرنامج الخاص بالمتابعة هو "

```
void Fibonacci (int n)
{ // Compute the nth Fibonacci number
  If (n<=1)                                +1
    Count<<n<<endl                          +1
  Else
  { int Fnum1=0,Fnum2=1,Fn                  n-2
    For(int i=2;i<=n;i++)                    n
    { Fn=Fnum1+Fnum2;                        n-1
      Fnum1=Fnum2;                          n-1
      Fnum2=Fn;                             n-1
    }
    Count<<Fn<<endl                          +1
  }
```

إن حاصلات هذه المتتالية تتصاف بالمتطير (n)

تصنيف الخوارزميات - تطبق الخوارزميات مع خانات حركته بخبر أقدم التفاضلات
(Fnum1, Fnum2, Fn, i, n) وعنوان العودة وهو خزن ثابت / يعده على حاصلات العمل
أي إلى

$$S_{Fibonacci}(n) = O$$

تعقيدات الزمن

بعد أن عتبر خوارزميات التحليل تعقيدات الخوارزميات وحلكه أو حركه شرط كالتالي .
الحالة الأولى - عند (n=0, n=1) في تعقيدات الزمن (عدد: تعقيدات) هي (2).
الحالة الثانية - عند (n=1) في عدد الحركات هو (4n+1) وكما يلي .

$$T_{\text{Algorithm}}(n) = \begin{cases} 2 & \text{if } n \in (0,1) \\ 4n+1 & \text{if } n \geq 1 \end{cases}$$

ولنعم بعبارة حساب عدد الخطوات في الاجراء البرمجية التالية

```
int i = 1
while (i <= n)
{
  x += 1;
  i += 1;
}
```

نلاحظ ان الرتبة (while) تأخذ (n-1) من الخطوات مع مراعاة الانسيب إلى بداية الحداث المستخدمة (i) والرابعة المعطاة له.

بما في جزء (Do while) هذا نلاحظ ان (while) والخطوات الحاصلة به فعدد (n) من الخطوات

```
int i = 1
do
{
  x += 1;
  i += 1;
} while (i <= n);
```

هنا : أوجد تعيينات التخزين و، توقف لتسليمه حساب ما يعنى بالبرمطاط السبقة (Prefix Average) لتتابعه من الإعداد

يفكر برصيح التسليم كالآتي: بما كان يجب مصفوفة معينة ولكن (X) مخصصة بحري (n) من الإعداد مصفوفة مثل الخطوات حساب مصفوفة معينة هي (A) حيث ان نحصل (A1) نفس البرمط قيم الحاصل من (X[0] ... X[i]) لقيم (i=0 1, ..., n-1) في انه

$$A[i] = \frac{\sum_{j=0}^i x[j]}{i+1}$$

Algorithm Prefix Averages(x).

Input An n-element array x of number

Output An n-element array A of number

That A[i] is the Average of elements X[0], ..., X[i]

1.1 تحليل لأفضل وأسى وأوسط حالة تنفيذ (Best & Worst & Average Casesse Analysis,

يجب علينا ان نلاحظ هذا إذا كانت المسألة تأخذ بكثر من حالة وسوف نقوم بالتركيز على الحالة الاسوأ للمساكن لأنها تحوي تعقيدات كثيرة، كما يمكننا التخلص من الصعوبات في الحالات التي تكون فيها التعقيدات المعنوية (مضامين معنوية)، غير مناسبة في كفاءة وحدها التحليل عدد الخطوات وذلك من خلال تحديد تلك أو مع الخطوات.

أو: عدد خطوات الحالة الافضل وهو التي عدد من الخطوات يمكن تنفيذها بفعالية عالية (أولى) عدد خطوات الحالة الاسوأ وهو أقصى عدد من الخطوات يمكن تنفيذها بفعالية معوية (أفضل) عدد خطوات الحالة الأوسطه هو العدد المتوسط من الخطوات التي يمكن تنفيذها على أنه مثله بفعالية معوية.

مثال: // يوجد العنصر الأكبر في مصفوفة بعينه العدد

Algorithm Arraymax (A,n)
Input An array A storing n integer
Output the maximum element in A

a[0] ← CurrentMax
1 to n-1 do ← For i
 If CurrentMax < A[i] then
 A[i] ← CurrentMax
 Endif
Endfor
Return CurrentMax

- الحالة الأفضل تكون البحث في أقصى حالاته عند تكون أول عنصر في المصفوفة هو الأكبر حيث لا يدخل في تنفيذ أي حل (if)
- الحالة الاسوأ يكون البحث في أسوأ حالاته عندما يكون أخر عنصر في المصفوفة هو الأكبر حيث سيتم تغيير القيمة حتى الوصول إلى النهاية
- الحالة الأوسطه: حيث تأخذ تعقيدات الحالة هي (عدد الخطوات لحد الوصول إلى العنصر (1,2,3, ... n) عدد الخطوات لكنه (n))

كما نلاحظ ان مستوى التعقيدات يكون حسب الحالة ففي الحالة الأفضل تكون أقل تعقيدات وفي الحالة الاسوأ تكون أعلى تعقيدات.

ملحوظة: // إلى طريقة حساب عدد الخطوات (Step Count) هي طريقة حساب تقريبية وليس دقيقة وهي صعبة نفس سرفا

$$T_{\text{Arithmetic}}^L(n) = 2n + 1$$

$$T_{\text{Arithmetic}}^R(n) = 3n$$

$$T_{\text{Arithmetic}}^A(n) = \frac{\sum_{i=1}^n (2n + i)}{n} = \frac{2n + \sum_{i=1}^n i}{n} = \frac{2n + \frac{n(n+1)}{2}}{n}$$

مع ملاحظة أنه في حاله المتوسطة فإنه يجري للعدد (i) إلى يبدأ من الصفر أو الواحد ، أما بد كلف عملية الحساب يبدأ بالعكس أي من نهاية إلى البداية فتكون

$$T_{\text{Arithmetic}}^A(n) = \frac{\sum_{i=1}^n (2n - i + 1)}{n}$$

5.1 أصبع التقريبية (Asymptotic notation)

يوجد ثلاث أصبع تقريبية هي

- 1 صيغة الحد الأعلى (Big-Ob)
- 2 صيغة الحد الأدنى (Omega)
- 3 صيغة الحد الأوسط - حد الثابت (Theta)

برهنت عدله تحدد عد الخطوات (Steps Count) على فني مهمة غشه في الصعوبة بذلك
لنحفظ إلى أصبع تقريبية لتحديد عد الخطوات

1 صيغة الحد الأعلى (Big-O)

ويقصد به إن تعقيدات الخوارزمية أو الوقت ممكن أن تساوي بعد الأعلى أو تكون أقل منه ولا يمكن أن تكون أعلى منه وعصية أصعب غه تتم كالآتي

$$f(n) \leq C \cdot g(n)$$

هذه المعادلة تطبق لنا فقط إن يوجد عدلان موجبان لها (C و n₀) بشرط أن
(C > 0 و n₀ > 1) حسب التعميد التالي

$$f(n) \leq Cg \quad n$$

نصعب لهم $n \geq n_0$

نظرية: لا كلف

$$f(n) = a_m n^m + a_{m-1} n^{m-1} + \dots + a_1 n_1 + a_0$$

فلنحدد حدود درجاتها (m) فلن -

$$f(n) = O(n^m)$$

وهذه بعض الأمثلة لتطبيق النظرية

مثال // ما كانت $3n + 2$ مثل $O(n)$ (تقنيات خوارزمية) لأي من تفرع معين في الحل ؟

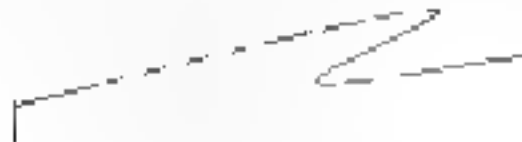
$$3n + 2 = O(n)$$

$$3n + 2 \leq 4n \quad \forall n \geq 2$$

بجميع قيم $n \geq 2$

حيث إن الكمية $(4n)$ مثل التواتر أي n هي ثابت $g(n)$ بـ 4 مثل للثابت c ، حيث إننا نختار أكبر معامل 4 وهو (3) وأصغره واحد ليصبح (4) ، كما في مثله n $O(n)$ مثل الحد الأعلى لتقنيات الخوارزمية .

وهذا يعني أنه يمكنه العمل التحصيل (النقاط الحقة) في منطقة معينة من دائرة مع التفرع على شكل الدالة بدون تأثير كما في الشكل رقم (2) التالي



شكل (2) منطقة Notation O

- تعنيه القسمة منطوية
- على الأقل يوجد عملية واحدة

ملاحظة // في الرموز مستخدم الساتر تبدأ من الإحداثيات $(0,0)$ إلى $MaxX,MaxY$ لأن سطر الحزب المخطط فقط .

Example1 Use Big O Notation to analyze the time efficiency of following c++ code of the integer N

```
For(int i=1; i<=N/2; i++)
{
    For(int j=1; j<=N*N; j++)
    {
        // ...
    }
}
```

المعقد بين دوائر For() الخارجية مثلا (N/2) بينما For() الداخلية مثلا (N*N)
 $N/2 * N * N$
 $\frac{1}{2} N * N^2 = \text{big-O} = O(N^3)$

EX N=5;
 for(int k=1; k<=2; k++)
 for(int j=1; j<=25; j++)
 { }

Then $O(125)$

k	J
1	1
	.
2	25
1	1
	.
2	25

Example2 Use Big-O Notation to analyze the time efficiency of following c++ code of the integer n

```
For(int i=1; i<=n/2; i++)
{ }
For(int j=1; j<= n*n; j++)
{---}
```

$$n^2 + n^2 n = 1/2n + n^2 \quad n + n^1$$

$$n(1+n) \quad \text{Big-O} = O(n^2)$$

ملاحظة/ المكدس (1/2) و (1) بهمانى ، عدل هيا كعدة باشه بيرى ليا (C) موصفة (Big-O)
 ان نختار الى فرقت ، فلو فرصد هتت

N=2;
 For(int i=1; i<=2; i++)
 For(int j=1; j<= n*n; j++)

هناك يحتاج (27) تكرار فقط للتكرار وهذا يعني إلى حالة السعيد هذا في أسوأ من الحالة السعيد
هناك لا تقرب من انه هناك المصعب التالي.

```

k=n
Do
{
  K=k/2
} While (k > 1);

```

هنا عندما تنقص قيمة n فهذا يعني انه الدالة $n \log n$ أو $\log n$ يتناقص بينما لا يمكن
أن يكون $O(n^2)$ ، $O(n)$ هي $O(n^2)$ ، $O(n^2)$

When $n=8$,
Then $k=8$,

k	الحقيقة
8	$8 > 1$
4	$4 > 1$
2	$2 > 1$
1	$1 \leq 1$

إن عدد مرات تكرار هذا المقطع ينقص إلى النصف في كل مرة هذا لأنه ينقسم إلى $\log n$
أي إلى (Big-O) له في $O(\log n)$ وسجده $O(n^2)$ ، $O(n)$ ، $O(n^2)$ ، $O(n^2)$ رده لها
تكرر من أقصاه وحدها $n \log n$ لأننا نصوب في n .

كما يمكن صياغته بشكل يتصوره التاليه
أيك المصنفه التالية

$3n + 2 \leq 4n$
المطلوب: إذا المصنفه التقريبية أي ثم إذا هذه (n) .

الحل: أي المصنفه يكون شكله كالتالي:

$$f(n) = a_m n^m + a_{m-1} n^{m-1} + \dots + a_1 n_1 + a_0$$

بعد من المصنفه فيها أول أو يساوي هذا يعني ان المصنفه في $O(n)$ (Big-O) المصنفه على
 n هوها إلى على أقصاه (n) هو الواحد وبالتالي يكون الحل المصنفه الأولى.

$$3n + 2 \leq 4n \quad O(n)$$

$$3n + 2 \leq 4n \quad \text{لا}$$

$$n \geq 2 \quad \text{بجميع قيم}$$

n	للطرف الأيسر	لتطرف الأيمن	$3n + 2 \leq 4n$	نتحقق
1	5	4	$4 \leq 5$	False
2	8	8	$8 \leq 8$	True
3	11	12	$12 \leq 11$	True
4	14	16	$16 \leq 14$	True

• الصيغة تحقق عندما قيمة الـ (n) أكبر من أو تساوي (2)

مثال 2: بنا كلف $10n^2 + 4n + 2$ يعش $P(n)$ (التخفيف الحرر ووقت) ليرتفع معنى
أوجد التعقيد بدلالة الصيغ التقريبية .

$$10n^2 + 4n + 2 = O(n^2)$$

$$10n^2 + 4n + 2 \leq 11n^2 \quad \forall n \geq 6$$

كما يمكن ملاحظة السؤال بالصورة التالية
سأنا نستخدم التالي

$$10n^2 + 4n + 2 \leq 11n^2$$

ف هي الصيغة المستعملة واقية لـ (n)

الحل// بعد أن الصيغة أقل أو تساوي من الصيغة هي $(Big-O)$ و على أن لـ (n) يعش أن
الـ (n) في الصيغة

$$10n^2 + 4n + 2 = O(n^2)$$

$$10n^2 + 4n + 2 \leq 11n^2$$

جميع قيم $n \geq 6$

التحقق	$10n^2 + 4n + 2 \leq 11n^2$	الطرف الأيمن	الطرف الأيسر	n
False	$1 \leq 16$	11	16	1
False	$44 \leq 90$	44	90	2
False	$99 \leq 104$	99	104	3
False	$176 \leq 178$	176	178	4
True	$275 \leq 272$	275	272	5
True	$396 \leq 286$	396	286	6
True	$847 \leq 786$	847	786	7

مثال 3: بنا كلف $(1 - 100)$ أوجد التعقيد بدلالة الصيغ التقريبية

الحل// بن قيمة 100 يعش $P(n)$ من التي $P(n)$ يمكن نقوله بالقيمة (1) ، فذا يعني

$$100 \leq 100 + 1$$

جميع قيم $n \geq 1$

مثال 4: $5 + 2^n + n^2 = O(2^n)$ أوجد تعقيد الحرر ووقت بدلالة الصيغ التقريبية

الحل// لاحظ أن هذا الترتيب هناك تعقيد أسية وهي على تعقيد ذلك إلى .

$$6 * 2^n + n^3 = O(2^n)$$

$$6 * 2^n + n^3 \leq 7 * 2^n \text{ لن } n \geq 4$$

نطبق قيم $n \geq 4$

ملاحظة: إذا وضعنا قيمة (n) بحيث تكون على n فموجود في الأمتة السعة فهو لا يؤثر
ويعتبر في باقي العلاقات صحيحة .

$$\text{مثال 5/ بعض من صحة المعادلة } 10n^2 + 4n + 2 \neq O(n)$$

نظن

$$10n^2 + 4n + 2 \neq O(n)$$

$$10n^2 + 4n + 2 \leq 10^4 n \text{ لن } n \geq 10^4$$

نطبق قيم $n \geq 10^4$ وهذا غير ممكن

$$\text{مثال 6/ بعض من صحة المعادلة } 3n + 2 \neq O(1)$$

نظن

$$3n + 2 \neq O(1)$$

$$3n + 2 \leq 10^4 * 1 \text{ لن } n \geq 10^4$$

نطبق قيم $n \geq 10^4$ وهذا غير ممكن

لأنه لا يمكن أن يكون $n \leq 10^4$.

2. صيغة أوميجا الكبرى (Omega Ω)

ويقصد بها أن عقيدت الحدود أو الوقت يمكن أن تكون كبير أو صغير أو متساوي لحد لآخر و
يمكن أن تكون أقل منه في نسبة الصعوبة يتم كتابتها .

$$F(n) = \Omega(g(n))$$

بما وفقد إذا كان لعدد ثابت موجب (C, n_0) بحيث $(C > 0)$ و $(n_0 \in \mathbb{N})$ لن

$$F(n) \geq Cg(n)$$

نطبق قيم $n \geq n_0$

نظرية: إذا كانت

$$f(n) = a_m n^m + a_{m-1} n^{m-1} + \dots + a_1 n + a_0$$

متعددة حدود من حيثها (n) فإن .

$$F(n) = \Omega(n^m)$$



شكل (3) بمسحه Notation Ω (يمكن)

- عملية التصاعد غير منقطعة (مستمرة)
- يوجد زاوية (0) وب قيمة
- ربما المعنى جيد عند 0 عند $[0, 2]$ وهناك استخدام أي دويقة خاصة سيحول أو يغير شكل الدالة تماماً

مثال: ليكن $3n + 2$ أقل من $3n$ ووقت البرهان معين أوجد هذه التطبيقات
بدلالة الصحيح القويبة ؟
الحل //

$$3n + 2 = \Omega n$$

$$3n + 2 \geq 3n \quad \forall$$

جميع قيم $n \geq 1$

لاحظ هنا مع n وهو (3) يعني كك هو أي إلى النأيب $c = 3$ ، كك يمكن بعين الصيغة
(1) $3n + 2 = \Omega n$ فهي لا تؤثر وتبقى الصيغة صحيحة

كك يمكن صياغة سؤال بالصورة التالية
نذكر المعطى التالي

$$3n + 2 \geq 3n$$

المطلوب: إيجاد الصيغة القويبة ثم إيجاد قيمة n

$$3n + 2 = \Omega 3n$$

$$3n + 2 \geq 3n \quad \forall$$

جميع قيم $n \geq 1$

n	الطرف الأيسر	الطرف الأيمن	$3n + 2 \geq 3n$	نتحقق
1	5	3	$5 \geq 3$	True
2	8	6	$8 \geq 6$	True
3	11	9	$11 \geq 9$	True

مثال 2/ // $\Omega(n) = 3n + 2$ ما هي المتحددة ؟

نفس تعريف المتحددة هي Ω هذا يعني ان العلاقة هي \geq فنكون المتحددة -

$$3n + 2 \geq 3n$$

مثال 3/ // نثبت لصيغة التقريبية التالية -

$$10n^2 + 4n + 2 = \Omega(n^2)$$

الخطوة 1// نكتب متحددة هذه الصيغة علماً ان $\{C=1\}$

تجرباً ، يف ان الصيغة هي الحد الأدنى هذا يعني ان الثابت يجب ان يكون أقل او مساوي للحد الأدنى اي 1

$$10n^2 + 4n + 2 \geq 10n^2$$

يصح قيم $n \geq 1$

مثال 4// // نثبت صحة هذه المعادلة $\Omega(2^n) = 6 + 2^n + n^3$

خطوة 1//

$$6 + 2^n + n^3 = \Omega(2^n)$$

$$6 + 2^n + n^3 \geq 6 + 2^n$$

لجميع قيم $n \geq 1$

3- صيغة الحد الأعلى Θ (Theta)

تستخدم الصيغة لتلخيص صلب المتحددة التي كتبت سابقاً

$$F(n) = \Theta(g(n))$$

إذا ربطنا إذا وحدث تتوافق العوحد التالية (C_1, C_2, C_3) نكتب نكتب

$$C_1 g(n) \leq F(n) \leq C_2 g(n)$$

لجميع قيم $n \geq n_0$

نظريه 1// إذا كانت

$$f(n) = a_n n^n + a_{n-1} n^{n-1} + \dots + a_1 n_1 + a_0$$

متحددة حدود مرتبتها (n) فن

$$F(n) = \Theta(n^n)$$



شكل (4): Θ Notation: (ثـ)

- كل زيادة تغير الشكل تماماً
- نفس الطول وكل مرة نحذف 10 ثوبه

مثال: أجب صحة المعادلة $3n + 2 = \Theta(n^2)$ بالعبارة بدلالة صحة مقاربهه:
الحل: المعادلة صحيحة

$$3n + 2 = \Theta(n^2)$$

$$\text{لأن } 3n \leq 3n + 2 \leq 4n$$

$$\text{بجميع قيم } n \geq 2$$

لاحظ هذا من قصد هذه المعادلة (2) يكون تصلي وليس عشوائي أي أنه (2) هو فرق تحقق
الصيغة لهذا القيمة (1) لا تحقق بالصيغة

مثال: على الصيغة التالية $3n \leq 3n + 2 \leq 4n$

الحل: لا فإنه يوجد غير على وفيه مظهر وهذا يعني أنه يجب استخدام صيغة Θ

$$n \log n = \Theta(n^2)$$

وبه في التواب (C) (C) أكبر من الصغر هذا يعني تحقق الشرط الأول سلك نقوم بتطبيق
الطريقة

التحقق	الطرف الأيمن	الطرف الأيسر	الرمز	n
False	3	5	4	1
True	6	8	8	2
True	9	11	12	3
True	21	14	16	4

سنتج في الحل الصحيح لك من $n \geq 2$

مثال 1/3: بين أن الصيغة التالية صحيحة

$$10n^2 + 4n + 2 = \Theta(n^2)$$

الحل: نبدأ من أعلى قيمة n نضرب بالتوالي

$$10n^2 \leq 10n^2 + 4n + 2 \leq 11n^2$$

لجميع $n \geq 5$

لاحظ أنه يجب أن نضع أعلى قيمة لـ (n) في الحدتين ونقفه الأكل لـ (C) نضع في النهاية اليسرى والأكبر في الجهة اليمنى

n	تضرب اليسر	الأوسط	تضرب اليمين	نتيجة
1	10	16	11	False
2	40	60	44	False
3	90	104	99	False
4	160	178	176	False
5	250	272	275	True

ملاحظة: في صيغة الحد n^2 على الأني هي للصيغة الأكثر دقة والتحقق عندما يكون $n(n)$ هو الحد الأعلى والأني للدالة $B(n)$.

نريد 1/4: برهن أن العلاقة التالية صحيحة

$$5n^2 - 6n = \Theta(n^2) \quad (C_1=5, C_2=11)$$

$$38n^2 + 4n^2 = \Omega(n^2) \quad (C=7)$$

نريد 1/2: برهن أن العلاقة التالية غير صحيحة

$$10n^2 + 9 = \Theta(n)$$

$$n^2 + \log_{10} n = \Theta(n)$$

وهذه بعض حلول الأمثلة السابقة بطريقة الصحيح الخوارزمية.

$$S_{\text{avg}}(a, b, c) = \Theta(1)$$

$$T_{\text{avg}}(a, b, c) = \Theta(1)$$

بداً إلى صيغة الحد الأعلى تساوي صيغة الحد الأدنى فيما يمكن تجنب استخدام صيغة التباين والعكس من ذلك وقد عد استخدام صيغة (Θ) وقد يمكن استخدام صيغة (O) أو (Ω)

$$S_{\text{avg}}(n) = \Theta(1)$$

$$T_{\text{avg}}(n) = \Theta(1)$$

$$\begin{aligned}
S_{\text{avg}}(n, m) &= \Theta(1) \\
T_{\text{avg}}(n, m) &= \Theta(m \cdot n) \\
T_{\text{avg}}(n, n) &= \Theta(n^2) \quad \text{وفي حالة } m=n
\end{aligned}$$

6-1 أوضاع التنفيذ (The Times Of Execution)

يمكن توصيف الوضع الذي يستطيع من خلاله تنفيذ وقت التنفيذ بالمعادلة التالية

$$O(1) < O(\log n) < O(n) < O(n \log n) < O(n^2) < O(n^3) < O(2^n)$$

- $O(1)$ تعني أفضل من باقي الوضع الأخرى وهذا بالتميز لبقاء الوضع هي $O(1)$ في أفضل من باقي الوضع التي بعضها
 - الخوارزميات التي لها تعقيد متزايد بوقت أكبر من $O(n \log n)$ بعد خوارزميات غير عملية وكذلك فإن الخوارزميات التي لها تعقيد ضيق أي $O(2^n)$ تكون ضيقة ولا تكون عملية إلا عندما تكون قيمة n صغيرة جداً أي أقل من 40
 - ولتوضيح ذلك نقرر من وجود حصة مقدار 10^9 نقطة ساعة الرقعة في وقت تكون
- $$P(n) = O(2^n)$$

When $n=10$ then $f(n)=1$ ms
 When $n=20$ then $f(n)=1$ ms
 When $n=30$ then $f(n)=1$ sec
 When $n=40$ then $f(n)=18.3$ min
 When $n=50$ then $f(n)=1.3$ day
 4×10^{14} year When $n=100$ then $f(n)=$
 32×10^{30} year When $n=1000$ then $f(n)=$

تعتبر معطيات، أوجد حل مسألة في وقتي باستخدام أسلوب التكرار (الزمن عام تقديري)

$$\text{Fibo}(n) = \begin{cases} n & n < 2 \\ \text{Fibo}(n-1) + \text{Fibo}(n-2) & \text{else} \end{cases}$$

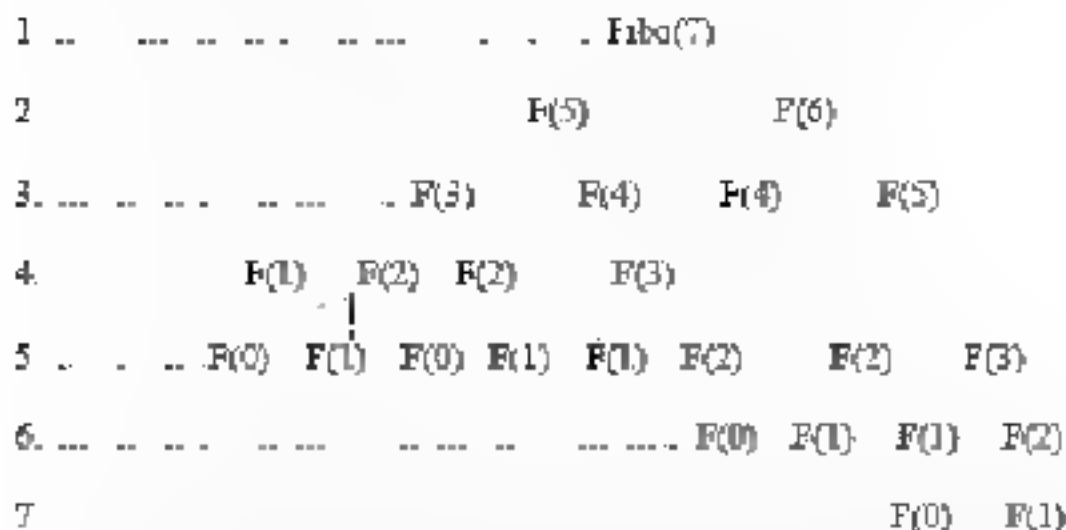
```

int Fibo1 (int n)
{
    if (n < 2) return n;
    Else
        Return (Fibo1(n-1)+Fibo1(n-2));
}

```

بحساب الوقت .

إن عملية حساب التكرارات بسلسلة ليست عام، فالتالي دائماً يحتاج إليها، إلى رسم مخطط تدويري يوضح تكرارات الاستدعاء، فالتالي يوضح رسم شجرة تنشيطات بسلسلة عدد فيبوناتشي كما في التالي:



1 - 2^k يمثل أقصى عدد للعقد في الشجرة، وبما أن كل عقد تمثل استدعاءً من هذه العملية تمثل عدد الاستدعاءات يقع بالخطوة k نحتاج إلى حساب التكرارات الخاصة بكل استدعاء.

7.1 الاستدعاءات (Recursion Tree).

ملاحظة: دعنا في الشجرة التكرارية أقصى عدد من العقد هو $(2^k - 1)$ حيث إن k يمثل عمق الشجرة (المستوى الأقصى) لذلك فإن تعقيد الوقت هو:

$$T_{Fibo}(n) = O(2^n)$$

وتوضيح ذلك:

عدد فيبوناتشي	عدد مرات تنشيط
7	1
6	1
5	2
4	3
3	5
2	8
1	13

نلاحظ أنه في كلتا حالتا $Fibo(30)$ فإن التنشيطات ستكون عدد هائل يقرب 500-000 تنشيط.

تعريف ١١: البحث خطي (أو بحث غير متوازي) باستخدام ماثرب (البحث بحث) تصبح تعقيدات الوقت خطية بدلاً من أسية.

مثال: نكتب بعضيات أقرب البحوث المتوازية والتوازي في ميثود البحث المتوازي (Sequential Search) التي تبحث في الميثود عن عنصر معين في مصفوفة بحاليد البحث بحث في هذه الحالة يمكن أن نرجع القيمة مبكر إذا كان العنصر غير موجود أو نرجع قيمة موقع العنصر إذا كان موجوداً.

```
int SeqSearch ( Type a[] , Type x , int n)
{
    int i=0;
    while (a[i] != x)
        i++;
    return i;
}
```

الحل ١١: نكتب تعقيد البحث المتوازي أي في العنصر الذي يبحث عنه موجود ضمن الفترة [1..n]

١. حالة الأفضل

$$T_{SeqSearch}^A(n) = \Theta(1)$$

٢. حالة الأسوأ

$$T_{SeqSearch}^W(n) = \Theta(n)$$

٣. حالة المتوسطة

$$\begin{aligned} T_{SeqSearch}^M(n) &= \frac{\sum_{i=1}^n (n-i+1)}{n} \\ &= \frac{(n+1)}{2} = \Theta(n) \\ &= \frac{1}{2} \cdot (n+1) = \frac{1}{2} \cdot n + \frac{1}{2} \quad \text{L 2} \end{aligned}$$

أما بالنسبة لخصائص البحوث المتوازية

$$T_{SeqSearch}^M(n) = \Theta(n)$$

ملاحظة: يمكن تطبيق أو استخدام ميثود البحث المتوازي للبحث عن عنصر في مصفوفة ثنائية الأبعاد وطبعاً سيكون هناك فرق من حالة إلى أخرى

تعريفات المعية (Practical Complexities)

في تعريف الوقت لبرنامج معين يكون وبصورة عامة في دالة يخصص المثال وهذه الحالة مفيدة جداً في تحديد كيفية تغير متطلبات الوقت بتغير خصائص المثال باستخدام دالة التعقيد الصافي مع زيادة برنامج تقريباً فالحل نفس المعية

ونفترض أنه لدينا البرنامج P يحوي تعقيداً رق في $\Theta(n^P)$ وبرنامج Q يحوي

تعقيداً رق في $\frac{Q}{\Theta(n^1)}$

السؤال هو أي البرنامج أفضل ؟

ولحل هذا نقول : علينا اتباع التالي

أولاً نكتب برنامج جديد بحيثما يكون بعد أعلى والأخر يكون أقل هذا يعني أن

من بعد البرنامج P الذي حجمه n على هو OM معية معية Q ولجميع قيم

$n_1 \geq n$ حيث n نقول (n) و Q نقول C

من بعد البرنامج Q الذي حجمه n على هو OM معية معية P ولجميع قيم

$n \geq n_1$

وحيث $n_1 \leq OM \leq n$ معية معية $n \geq \frac{Q}{P}$

في البرنامج P يكون مربع من البرنامج Q معية

$$n \geq \max \left\{ \frac{\alpha}{\beta}, n_1, n_2 \right\}$$

فإذا فرضنا أن البرنامج P بعداً في 10^4 على فاجبة بينما البرنامج Q بعداً في n^2 معية يكون

$$n \leq 10^4$$

$$M^2 = \frac{t - b}{n}$$

وفي حالة أنها أصبحت تربيعية فهذا يحصل قطع مكافئ:

$$t = a_0 + a_1 n + a_2 n^2$$

أما إذا كانت الخطية فهي $O(nm)$ لأنه يحصل محطاً ذو صفحة

$$t = a_0 + a_1 n + a_2 n \log n$$

مثال: لقرص قبض، اجريه سوا حلقه مخوار رمية يجب التعقيد (Sequential Search)

//منتج

إلا أن هدف من تجريبه هو قياس الحلقه الاسمى بحيث إن هذه سوار رمية تحتاج وقت قليل جداً راقب في الحاسوب هو أقل من جريء من الثانية لذلك يجب إن نعدّ دوائر برمانه الوقت كح في جزء البرمجه التالي

```
#include <iostream.h>
#include <iomanip.h>
#include <time.h>
void time Search()
{ // repetition factors
  Long int r[21]= {0,200,000,200,000,1000,000, . . . 25000};
  Int a[1001],n[21];
  For (int j=1;j<=1000;j++) a[j]=j;
  For (int i=1;i<=10;i++)
  { n[i]=10*(i-1);
    n[i+10]=100*i;
  }
  Cout<<" n \t t "<<endl<<endl;
  Cout<<Setprecision(6);
  For (int j=1;j<=20;j++)
  { int h=Gettime();
    For (int i=1;i<=a[j];i++)
    { k=SeqSearch(a,0,n[j]);
      Int h1=Gettime();
      Int t1=h1-h;
      Float t=t1;
      t*=r[j];
      cout<<Setw(5)<<n[j]<<Setw(5)<<t1<<Setw(8)<<" "<<endl;
    }
    cout<<" time are in millisecond "<<endl;
  }
}
```

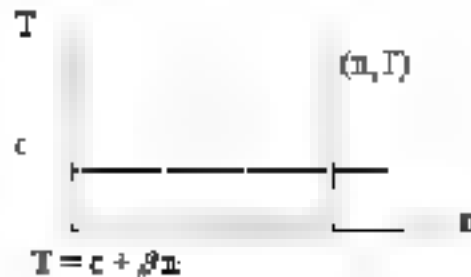
ويفيد بلى بوضوح نموذج التبريد.

- المصفوفة $[2][2]$ مستخدم لحزن قيم التكرارات التي تدور من 1 إلى 20 قيمة أي ألف موزن يُحدد 20 قيمة للـ n وهي غير خالية حيث إن القيمة من هذه التكرارات هي جعل الوقت أقل من الثاني. مع ملاحظة أنه كلما قادت n من هناك الوقت يقل. وبذلك يجب تأكيد التكرارات
- إن شكل قيمة في مصفوفة الـ n تكرار يندرج في مصفوفة الـ r وحجم مصفوفة الـ n هو 2 حيث أننا نضعي العناصر التي نبحث عنها والتي نعالقها في مصفوفة الـ r بالمصفوفة n بالمصفوفة g فهي تحوي العناصر التي نبحث فيها ليدها عن العناصر المطلوب
- إن العلاقة التي نحصل من مصفوفة الـ n تقدم هي معادلة حساب قيمة n وبذلك نطيرها من معادله إلى تحري.
- إن الدالة $Gettime()$ هي جزء آخر مني يجد الوقت الحقيقي للتحسين n متسلسلاً بجسراً
- منوه من القيمة هذا أن للقيمة في 10 هذا يعني على أنه لو كانت القيمة 100 لكانت دس
- السطر n الذي خصصناه للتحقق من القيمة n المرحلة صغرنا معطى الدالة
- يعني أي عنصر غير موجود بالمصفوفة n من قيمه يختلف عن قيمه ويحوي الفرق في الوقت.
- الدالة $Setprecision(6)$ تحوي خصيصاً بغير ثبات بعد الفاصلة بالعدد العشري n بالـ $Setn(5)$ فهي تحوي الانتقال منسلفاً بحجم الرقم المقصد صغرنا سطر الحلق.

إن نتيجة هذا البرنامج هي الحصول على وقت البحث (Sequential Search) مصفاً إليه وهو حواف التكرارات والنقص من وقت دور التكرارات عند إن حدد نفس الشجرة بالاصلاح إلى صفه جسم الشجرة أي لنا جعل جسم الشجرة فرعاً (for) ويسجل لنا من الذي بلغه كل دورة ثم بطرحه من قيم n في التجربة السابقة (قيمة وقت الدورة للوحدة) فطرح من كل قيم T

في التجربة السابقة كان وقت الدورة الواحد تقريباً (0.002) ولكنه غير ثابت في أنه قد يكون n من يكثير في التحسينات فذلك السرعة عليه

وهناك علاقة تربط حجم التكرارات (n) بالزمن (T) وهي علاقة الزمن بالحجم في البحث Sequential Search



شكل (6) علاقة الزمن بالحجم في البحث Sequential Search

الفصل الثاني

الترتيب

(Sorting)

2-3 حوارات مرتبة (Sorting Algorithms)

الخوارزمية هي عبارة عن مجموعة من خطوات الحسابية و الرياضية و المنطقية المستخدمة لحل مشكلة ما ، و سيتم الحواراتية بهذا الاسم نسبة إلى التحرك التسمي "بوجتر" مصطلح هو من الحواراتية ."

حواراتية ترتب هي حواراتية تكون من تقسيم مجموعة عناصر حسب ترتيب محدد العناصر الفريدة ترتيب، يوجد في مجموعة مرتبة بعلاقة ترتيب معينة

تصنيف حواراتية الترتيب مهم جداً لأنه يمكن من اختيار نوع الخوارزمية الأكثر مناسبة للشكل المطلوب مع الأخذ بعين الاعتبار الترتيب الموجودة في الحواراتية

بمعنى آخر الترتيب عبارة عن عملية ترتيب مجموعة من العناصر الترتيبية والتي قيمته معينة سمي حقل أو حق حقول تسمى مفتاح أو بصوره خاصية أو تسمية الخواص من الترتيب هو

1 - رتبة كتابة الخوارزمية " يجب عن عناصرها "

مثال // تمثيل الأرقام 3 و 4

3	4
النظام العشري	النظام العشري
11	100
النظام الثنائي	النظام الثنائي

0 0 0 0 1 1

0 0 0 0 1 0 0

استخدام 2 بيت للحي

0 0 0 0 1 1

استخدام بيت 1

حسب في المساحة الحرة لأن لا صفر تسجل موقع

2 - تبسيط معالجة الملفات

لأن الملفات تتألف من حقول من ترتيب هذه الملفات حسب مفتاح يكون تسجل في حصة الترتيب من البحث .

مثال // هناك بيت من ثلاث حقل هو تسجل "الطالب" و"التي" اسمه و"النتيجة" العدد ، ستصبح أن نستخرج المعين للترتيب الأسماء حسب الشخص و"النتيجة"

3 - حل مشكلة مشابه الفريد

يوجد مشكلة في الفريد هي مشابه الأسماء إذا كان الاسم مشابه لا يمكن أن يأخذ اسم الأب وإذا كان اسم مشابه تلحق اسم الجد ، مثل اسم ريتاب

ريتاب حيدر محمد

ريتاب حيدر محمد

2-2 أنواع الترتيب (Types of Sorting)

- 1 الترتيب الداخلي (Internal Sort)
- 2 الترتيب الخارجي (External Sort)
- * الترتيب الداخلي يجب ان تكون الذاكرة بحيث يكون حجم البيانات مناسب وليس كبير. ويشمل:
 - 1- ترتيب الاختيار (Selection Sort)
 - 2- ترتيب الفقاعي (Bubble Sort)
 - 3- ترتيب الإدخال (Insertion Sort)
 - 4- ترتيب قنبر (Shell Sort)
 - 5- الترتيب السريع (Quick Sort)
 - 6- ترتيب الأسفل (Radix Sort)
 - 7- ترتيب المؤشرات (Pomiers Sort)
 - 8- الترتيب الشجري (شجرة البحث الثنائية) (Tree Sort)
 - 9- Topological sorting

* الترتيب الخارجي هو الترتيب الذي يجب ان يخرج الذاكرة الى اوسط التخزين التقوي عند يكون حجم البيانات كبير بحيث يصعب استيعابها في الذاكرة. هذه عملية الترتيب ويشمل:

- 1 الترتيب بالدمج (Merge Sort)
- 2 الترتيب بالدمج المتوازن ذو السلاسل (Balanced Two Way Merge Sort)
- 3 الترتيب بالدمج باستخدام طريقة قسم وانضم (Divided & Conquer Merge Sort)

تعتبر الترتيبات المعقدة لا تفضل حواريه الترتيب.

- 1 حجم البيانات المطلوبة ان كان صغير يكون ترتيب داخلي أما ان كان كبير يكون الترتيب الخارجي.
- 2 نوع التخزين ان كان دائري ويستخدم يكون الترتيب داخلي أما ان كان خطي فالترتيب يكون الترتيب الخارجي.
- 3 درجة ترتيب البيانات حيث ان ترتيبات الترتيب مرتبة لترتيب بشكل أسرع من الترتيبات غير المتعددة الخطوات.

2 3 خوارزميات الترتيب الداخلي (Internal Sort):

1 خوارزمية الاختيار (Selection Algorithm)

ترتيب الاختيار هو خوارزمية لترتيب الأكثر بديهية ، و يتم عن طريق البحث في كل عنصر الأكبر من كل العناصر الأصغر ، و يتم وضع في المكان الأخير ثم يبحث عن ثاني أكبر ، و يتم وضع في مكانه في كل العناصر الأصغر ، و يتم وضع في الأخير ، حتى يتم ترتيب الجدول بالكامل.

تصانص برتيب جدول هـ

- 1 عند التقاطع في الصفه برتيب جدول عند عنصره V هو $2(N-1)$
- 2 عند التقاطع في رتيبه N

ويمكن ان يصيح تلك تصانص الصفه الآتيه

- 1 تصانص تصانص في الصفه واستثنائه من موقعه مع التصانص في الموقع V في الصفه
- 2 تصانص تصانص من التصانص في الصفه و استثنائه من موقعه مع الموقع الثاني في الصفه
- 3 تصانص هذه الصفه حتى الوصول إلى التصانص الأول

مثال: رتب التصانص في الصفه باستخدام طريقة الاختيار (Selection Algorithm)

8 9 7 6 4

Index	1	2	3	4	5	6
8	1	2	2	2	2	2
9	1	3	3	3	3	3
7	1	4	4	4	4	4
6	1	5	5	5	5	5
4	1	6	6	6	6	6
3	1	7	7	7	7	7
2	1	8	8	8	8	8
1	1	9	9	9	9	9

النتائج:

عدد التصانص $N=7$

عدد المراحل $N-1=6$

ملاحظة: معدل المقارنات هي $2(N-1) * N$ حسب

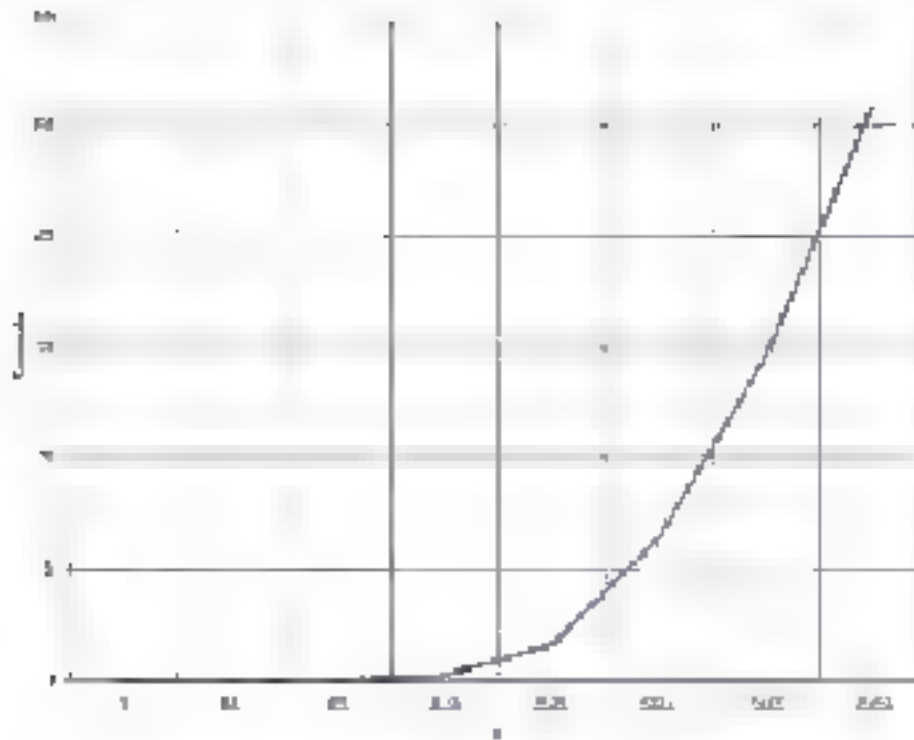
$$2 * 6 * 7 = 84$$

ملاحظة: عدد المقارنات بعدد على عدد المراحل ونساقص في كل مرحلة من حيث إلى أن
نصل إلى المرحله الأولى عدد المقارنات التي عرفت بها هي $N-1=6$ ونحن التقديرات هو 31،
تتمكنه في طريقة الاختيار التي لاحظناه في هذه المرحله من كل عنصر يهبط إلى
موقعه في كل مرة لتتعلق لأنه يمكن أن نصل إلى موقعه.

مثال: تصانص صفه هـ ثلاث عناصر مثلا التصانص الذي هغه (3) كال في الموقع (2) وتصانص
في الموقع (1) هما التصانص (8) في الموقع الثاني يعني في بعض المواقع ويذكر عن تلك التصانص
إلى مرحلة تصانص أنه يجب أن يقرن مع الرقم (9)

8	3	3
3	8	8
9	9	9

التحليل التجريبي (Empirical Analysis)



شكل (7) العلاقة بين حجم المدخلات (n) ووقت التنفيذ (T) لخوارزمية فرز البند (Selection Sort Efficiency).

والدالة الخوارزمية التي تقوم بتطبيق هذه الطريقة هي

```
void selectionSort(int numbers[], int array_size)
{
    int i, j;
    int min, temp;
    for (i = 0; i < array_size - 1; i++)
    {
        min = i;
        for (j = i + 1; j < array_size; j++)
        {
            if (numbers[j] < numbers[min])
                min = j;
        }
        temp = numbers[i];
        numbers[i] = numbers[min];
        numbers[min] = temp;
    }
}
```

هذه، دراجع يقوم باستخدام طريقة الترتيب بالاختيار لمجموعة كلمات

```
#include< stdio.h>
main( )
{ Char *z,
  Char *name[] {"ammar", "Fanna", "omar" "shameel", "jamal",
  "saeed", "yousef", "marham"},
  Int nmax=8
  Register int i,j,k;
  For i=0; i<=nmax-1 ++i)
  { k=
    z=name[i],
    For(j=i+1, j<=nmax; ++j )
    {if (strcmp(name[j],z)< 0)
      { k=
        z=name[j],
      }
    }
    Name[k]=name[i];
    Name [i]=z,
  }
  For(i=0; i<=nmax; ++i)
    printf("%s<n" name[i])
}
Ahmed
Ammar
Fanna
jamal
Mariam
Omar
Saeed
yousef
```

2. خوارزمية الترتيب الفقاعي (Bubble Sort Algorithm)

ترتيب العناصر خوارزمية ترتيب بسيطة لطيفة، وهي تعمل على وضع العنصر الأكبر كفقاعة الهواء التي ترتفع إلى أعلى وذلك بترتيب العناصر بتتابع. أي نقوم بمقارنة العنصرين الأول والثاني، نحفظ العنصر الأكبر، و نبدل الأماكن إذا كانا غير مرتبين. نقوم بهذه العملية إلى آخر عنصر، بعد ذلك نعيد ترتيب باقي العناصر إلى الأخر وهكذا نواصل حتى نصلح عدد وجود عنصر بالحد 1 في صفح لا نقوم بالتبديلات بعد بمر عليه

$$\frac{n(n-1)}{2}$$

لترتيب جثث A بعدد n ، فإن عدد المقارنات سيكون:

$$\frac{n(n-1)}{2}$$

كما عدد التحولات فهو في المتوسط:

تقوم هذه الخوارزمية بترتيب عناصر n قسماً ثنائياً على عدة مراحل عددها $\log_2 n$ ، حيث يتم وضع عدد واحد على الأقل في رتبة الصحيح بنهاية كل مرحلة.

خطوات تطبيق الخوارزمية:

- 1- إدخال الأعداد المراد ترتيبها في مصفوفة X .
- 2- استخدام متحول $switched$ بوضعته على صورت ($switched=FALSE$) أو عدم حدوث "تغيير" لتعطي ($switched=TRUE$).
- 3- استخدام حلقة خارجية بعدد المراحل $n-1$ ، حيث سوف الحلقة في حال عدم حدوث تبديل (أو عند هزئية).
- 4- استخدام حلقة داخلية تقارن كل عدد بالعدد الذي يليه، حيث يتم تغيير قيمة المتحول $switched$ إلى $true$ في حال التبدل.
- 5- استخدام حلقة لظهور ترتيب الأعداد في نهاية كل مرحلة.
- 6- استخدام حلقة لإظهار ترتيب الأعداد النهائي.

وهذا جزء البرنامج بالتطبيق الحاصل:

```
#include <iostream.h>
#define MAXNUM 10
enum boolean {FALSE, TRUE},
void main()
{int X[MAXNUM]
  int n, pass, hold;
  int switched=TRUE;
  cout<<"Enter count of numbers:"
  cin>>n;
  for(i=0; i<n; i++)
    cin>>X[i];
  for(pass=0; pass<n-1 && switched==TRUE; pass++)
  { switched=FALSE;
    for(i=0; i<n-1-pass; i++)
      if(X[i]>X[i+1])
      { switched=TRUE;
        hold=X[i];
        X[i]=X[i+1];
        X[i+1]=hold;
      }
  }
```

```

for(i=0; i<n; i++)
    cout<<"X["<<i<<"]="<<endl;
    cout<<endl;
}
cout<<"The sort is: "n<
for(j=0; j<n; j++)
    cout<<"X["<<j<<"]="<<endl;
}

```

نذكره هذه الطريقة لتصنيف أيجاد أصغر العنصر الموجود في قائمة ما، ثم نضعه في مكانه الأول، ونكرر العملية حتى نحصل على القائمة مرتبة.

- 1- First pass
- 2- Second pass

خطى عمل هذه الطريقة هي:

- 1- نأخذ العنصر الأول في القائمة ونضعه في مكانه الأول.
- 2- نأخذ العنصر الثاني في القائمة ونضعه في مكانه الثاني.
- 3- نكرر العملية حتى نحصل على القائمة مرتبة.

مثال: ترتيب العناصر 8, 3, 9, 7, 2 بطريقة الترتيب الفقاعي (Bubble Sort Algorithm).

List	pass(1)	pass (2)	pass (3)	pass (4)
8	8 3 9 7 2	2 2 2	2 2	2
3	3 3 2 8	8 8 3	3 3	3
9	9 2 3 3	3 3 8	8 7	7
7	2 9 9 9	7 7 7	7 8	8
2	7 7 7 7	9 9 9	9 9	9

عدد العناصر $N = 5$
 عدد التمريرات $N - 1 = 4$
 عدد المقارنات $N^2/2 = 25/2 = 12.5$
 عدد التبادلات $N^2/4 = 25/4 = 6.25$

من هذه الطريقة تكون جيدة إذا كانت العناصر متباعدة عن بعضها وعندئذ ينشأ كثير من المقارنات إلى مساحة تخزين كبيرة، بهذا فنحن نقف امام هذه الخوارزمية $O(N^2)$

3 خوارزمية الإدخال (Inserting Sort Algorithm)

نأخذ نفس هذه الخوارزمية كما يلي

1. نبدأ بالعنصر 2 في القائمة ونقارنه مع العنصر الأول ونضعه حيث الترتيب في مقدمته القائمة وبذلك ترتيباً تصاعدياً.
2. نبدأ بالعنصر 3 ونقارنه مع مقدمته القائمة فنرى بحسب ما على العنصر الأول والثاني ونضعه في الموضع التالي ونسهر بالتسوية حتى الوصول على لقائمة مرتبة

مثال: نأخذ العناصر الآتية بطريقة ترتيب الإدخال (Inserting Sort Algorithm)

8 3 9 7 2 6 4

الحل: يمكن ترتيبها كما في الجدول التالي

List	1	2	3	4	5	6
8	3	3	3	2	7	2
3	8	8	7	3	3	3
9	9	9	8	7	6	4
7	7	7	9	8	7	6
2	2	2	7	9	8	7
6	6	6	6	6	6	9
4	4	4	4	4	4	9

من الملاحظ الإدخال هو عكس ترتيب الاختيار لأنه يُنصّب العنصر ويقارنه مع العنصر الذي قبله حيث نأخذ الأول مع التالي والأول مع التالي وهكذا

عدد العناصر $N=7$

عدد المراحل هو $N-1=6$

مجموع عدد المقارنات هو $N^2=49$

مجموع التبادلات هو $N^2=49$

عقل // البرنامج يقوم بتصنيف حروف هجاء برتتيب الإصافه

```
typedef int tab_entiers[MAX];

void insertion(tab_entiers t) {
    /* Specifications extens */
    int i, p, x;
    for(i = 1 ; i < MAX ; i++) {
        /* position dissertation */
        /* determine p : 0 <= p <= i */
        /* t[p] <= t[i] */
        p = 0;
        while(t[p] > t[i]) p++;
        x = t[i]; /* t[i] */
        for(j = i-1 ; j >= p ; j--) t[j+1] = t[j];
        /* translation t[p+1] vers t[p+1+1] */
        t[p+1] = x; /* insertion t[i] */
    }
}
```

هذا // القائمة بصوي مجموعه عناصر ، العناصر المتفاله بالرمادي هي العناصر المتضره او
المتحنيه والتي ير . تر ايدها ، هذه العناصر المكتوبه بالقطب الحريص bold هي العناصر العربيه
في مكانها الصحيح.

ت. 13 - 14 - 10 - 29 - 37

29	10	14	37	13
----	----	----	----	----

الحل . يمكن توضيحه بالخطوات التاليه:

ت. 13 - 14 - 10 - 29 - 37

29	10	14	13	37
----	----	----	----	----

ت. 13 - 14 - 10 - 29 - 37

13	10	14	29	37
----	----	----	----	----

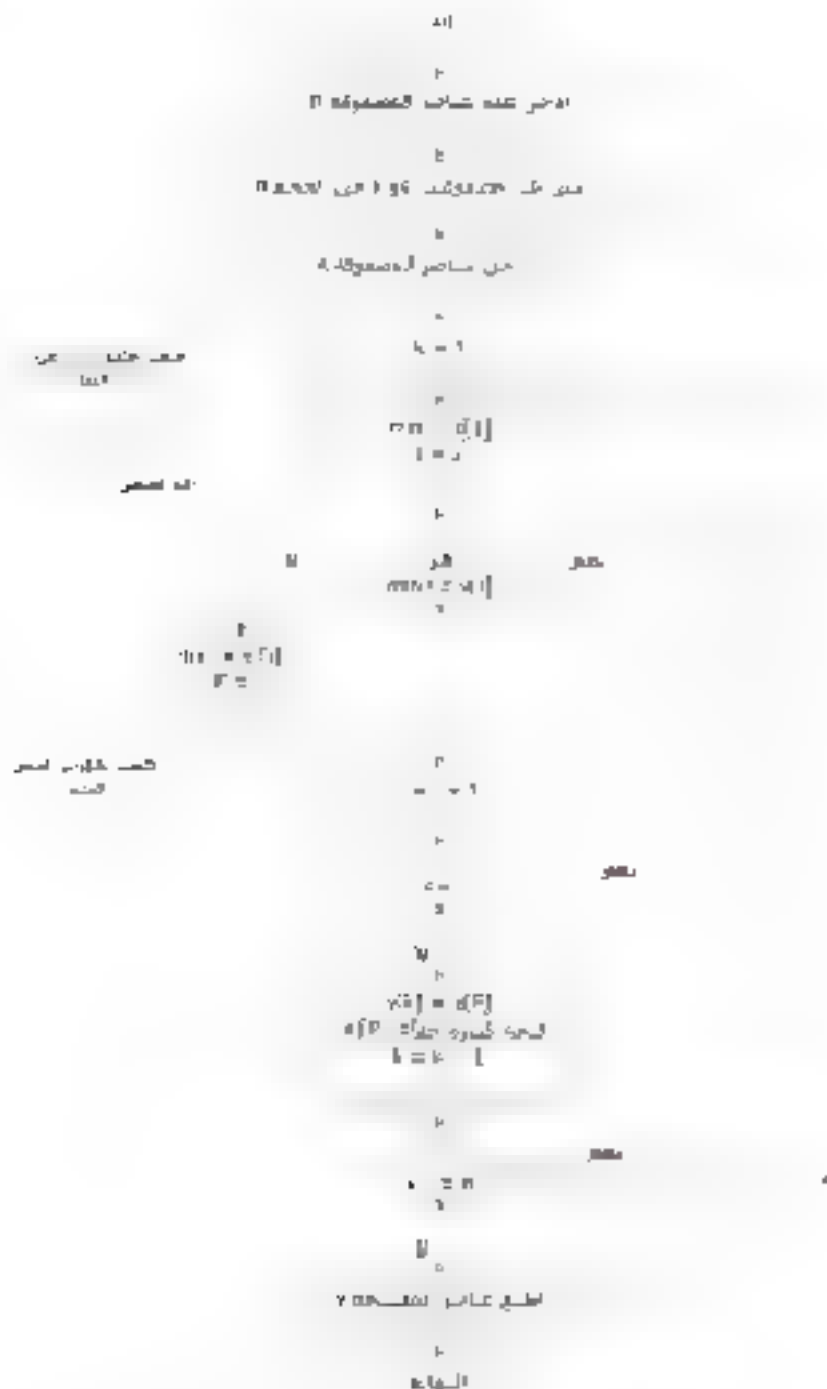
ت. 13 - 14 - 10 - 29 - 37

3	10	14	29	37
---	----	----	----	----

ت. 13 - 14 - 10 - 29 - 37

10	13	14	29	37
----	----	----	----	----

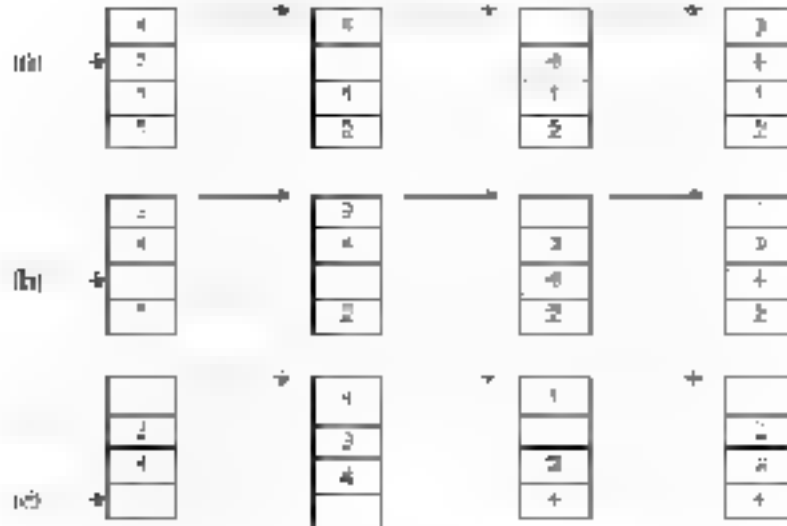
من الممكن ان تستخدم مصفوفتين بعدد عناصر بحيث تحتوي الاولى على عناصر غير صفريه ويتم تحويل هذه العناصر فترتيب بعدد حتى لا تقاربى الى الصفرية الثالثة ويمكن كتابة flowchart باستخدام مصفوفتين كما نرى



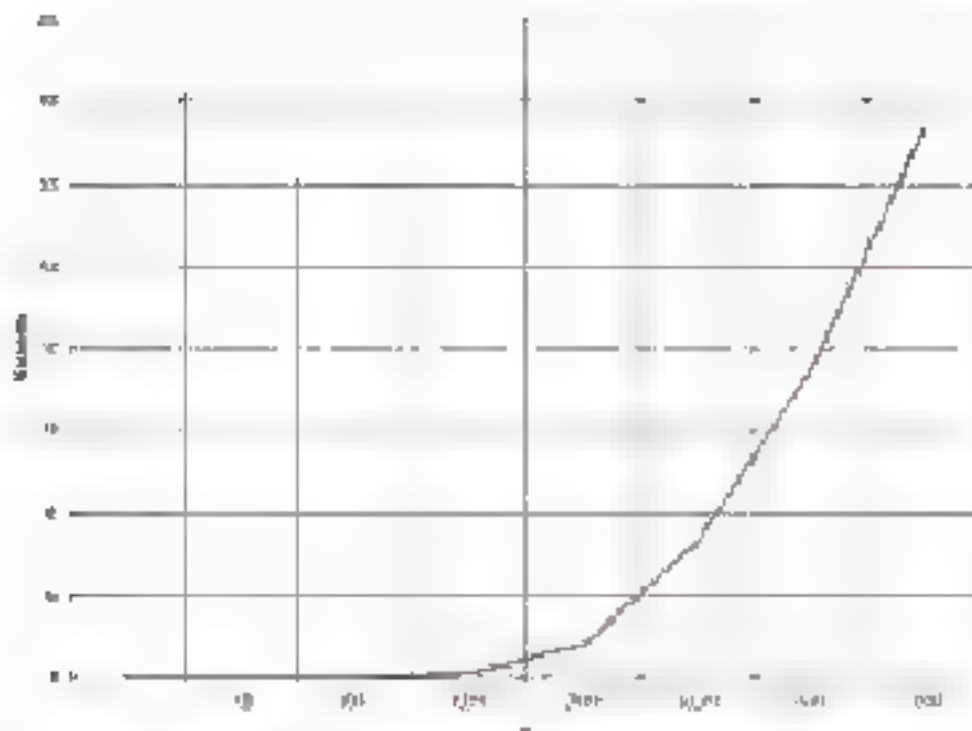
شكل (8) مخطط التدفق، يوضح فكرة خوارزمية الإضافة

مثال: ديف عنصر الثمانية رتبة (4,3,1,2) باستخدام خوارزمية البصالة .

الخطوة يمكن ترتيبها كما في الخطوات (a,b,c) التالية



تحليل تجريبي (Empirical Analysis)



شكل (9) كفاءة خوارزمية البصالة (Insertion Sort Efficiency)

والحقبة البرمجية الخاصة بتطبيق خوارزمية الإدخال هي :

```
void insertionSort(int numbers[], int array_size)
{
    int i, j, index;

    for (i=1; i < array_size; i++)
    {
        index = numbers[i];
        j=i;
        while ((j > 0) && (numbers[j-1] > index))
        {
            numbers[j] = numbers[j-1]
            j = j-1;
        }
        numbers[j] = index;
    }
}
```

مثال ١٢: برنامج يقوم باستخدام خوارزمية بربسب لإزالة التكرار من مجموعة بيانات

```
#include<iostream.h>
using namespace std;
{
    Char *z;
    Char *name[ ]={'ammar' "Fatima" "omar" "ahmed" "jamel",
                  "saeed" "yousof" "mariam"};

    Int nmax=8;
    Register int i,j;
    For (i=0;i<=nmax-1; ++i){
        z=name[i],
        j= 1;
        While (j>=0 &&(strcmp(z,name[j])<0)){
            Name[j+1]=name[j]
            j
        }
        name[j+1]=z.
    }

    For (i=0, i<=nmax, ++i cout<<" a"<< name[i];
}
```

Ahmed
 Ammar
 Fatma
 jassal
 Mariam
 Omar
 Saeed
 yousef

3 خوارزمية شيل (Shell Sort Algorithm):

يوجد مشكلة في الخوارزمية الباعثي هي (في عدد العناصر بزيادة لكن، عنصر في زيادة عدد العناصر في الأعداد في الخوارزمية) هناك أن تكون العناصر في بعد ترتيب (في آخر عنصر في الخوارزمية) وأن الموضع الصحيح يجب أن يكون في الموضع الأول، هناك أنه في عدد كبير من المقارنات وهذا يؤدي إلى كثرة الأخطاء.

ولحل لهذه المشكلة من خلال خوارزمتين هي:

1- خوارزمية شيل

2 خوارزمية الترتيب السريع

لكنها تقلص كلاً من. حذف بعض المتقدم الخوارزمية إلى مسافة واحدة، وتحتوي على عنصرين أو أكثر ليس متجاورين وإنما على نفس المسافة لنفسه، ثم ينقسم المسافة الأولية إلى النصف ويحتوي المقارنة أن تستمر ثلاثة إلى أن تصبح المسافة مساوية لـ 1، وبذلك يتم ترتيب الخوارزمية

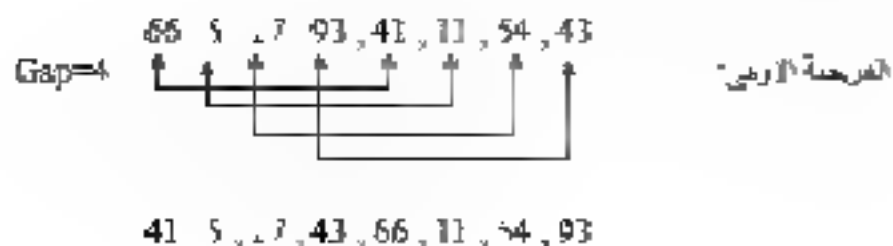
في المسافة الأولية بين عنصرين تدعى فجوة (Gap):

مثلاً: هناك 8 عناصر أولية 66, 5, 17, 93, 41, 11, 54, 43

الحل //

1 بحسب الأعداد المراد ترتيبها $N=8$

2 نضع المسافة الأولية في النصف 4 Gap



المرحلة الثالثة Gap 4/2 2

41 5 , 17 , 43 , 66 , 11 54 , 93



17 , 5 41 , 43 , 66 , 11 , 54 , 93



17 , 5 41 , 11 , 66 43 , 54 93



17 , 5 41 11 , 54 , 43 , 66 , 93

المرحلة الثالثة Gap 2/2 1

في هذه المرحلة نستمر بعملية السبيل حتى نحصل على القائمة مرتبة الإرتبة

5 11 17 41 , 43 , 54 , 66 , 93

خصائص تطبيق خوارزمية شل (Shell Sort Algorithm)

- 1 - يزيد كفاءتها كلما زادت عدد العنود
- 2 - لا بعد ج إلى مكان أصلي في القائمة لأجراء عملية الترتيب
- 3 - كفاءة إذا كانت القيمة داخل القائمة مرتبة أو شبه مرتبة ولتطلب نقل قيمة قليلة
- 1 - يزيد كفاءتها لأنه يتم ترتيب العناصر قبل الوصول إلى {Gap}
- 2 - وذلك لأنه يمكن أن تكون النتيجة أن يكون عنصر الواحد سور استخدام مكان خاص له
- 3 - وذلك لأنه لا يوجد في البداية بالآلية إذا كانت مرتبة أو عند الترتيب أقل من كفاءة الخوارزمية شبه مرتبة

تتميز خاصة تنظيم خوارزمية شل (Shell Sort Algorithm)

تستخدم هذه الخوارزمية للحصول على الحالة الأكثر ترتيباً في حالة الخوارزمية الكبيرة

تقدير الأول لتقدير معدل Gap حيث $Gap = 1.72 \cdot (N^{1/3})$

في المثال السابق استخدم

$$N = 8$$

$$4.5355555555555555 \approx 5$$

المعيار الثاني - تقدير أقصى قيمة لمعدل الوقت (Time average) حيث -

$$T_{avg} = N^2 \cdot \left(\frac{5}{3} \right) \cdot \frac{1}{8^{(1/3)}} \approx 32$$

مثال: ارباب عناصر القائمة (3, 5, 1, 2, 4) بطريقة شبيهة باستخدام فجوة عدد (2) مرة واحدة (1) ؟

الحل: يمكن ترتيبها كما في خطوات الآتية:

(a) $Gap=2$

3	5	1	2	4
4	3	5	1	2

1, 2 3 5 4

نفس الفكرة فكل مرة $Gap=1$

1	2	3	5	4
4	3	5	1	2

1, 2, 3, 4, 5

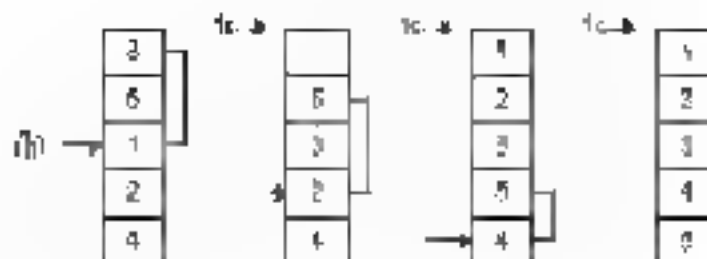
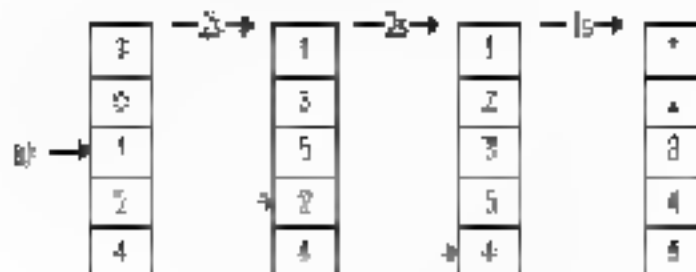
(b) $Gap=1$

3	5	1	2	4
4	3	5	1	2

نفس الفكرة التتبع حتى فصل كل قائمة من القائمة

1, 2 3, 4, 5

ويمكن تعديل القائمة من بعض عناصر القائمة بنفسه للعنصر الأصغر منه سي .

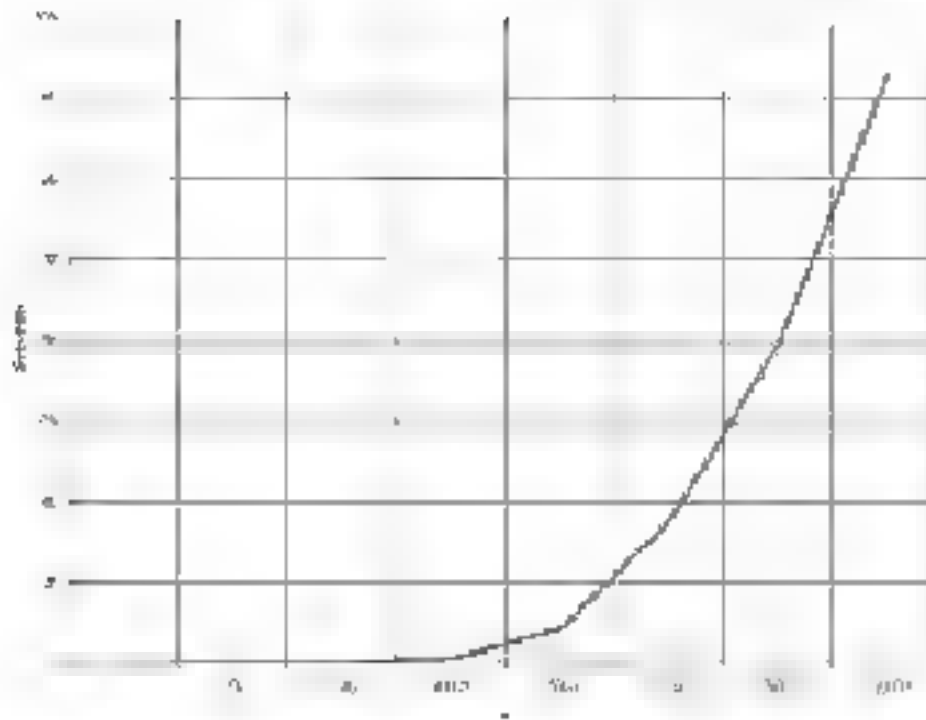


مثلاً: في المثال التالي يستخدم خوارزمية ترتيب شيل (shell) ترتيب مجموعة الأسماء

```
#include< stdio.h>
main()
{
    Char *name[ ]={ 'anmar' "Fatima", "omar" "ahmed", 'jamaal'
                    "saeed" "yousef", "mariam"},
    Int m[ ]={9,5,3,2,1},
    Int nmax=8;
    Register int a,b,c,t,v;
    Char *z;
    For (v=0;v<=5; ++v){
        c=m[v];
        t=c;
        for (a=t; a<nmax; ++a){
            z=name[a];
            b=a-c;
        }
        If(b==0){
            t=t;
            t++;
            name[t]=z;
        }
        While((strcmp(z,name[b]))<0)&&(b>=0)&&(b<nmax){
            name[b+c]=name[b];
            b=b-c;
        }
        name[b+c]=z;
    }
    For(a=0; a<nmax; ++a)printf("%s\n", name[a],
}
```

Ahmed
Anmar
Fatma
jamaal
Mariam
Omar
Saeed
yousef

التحليل التجريبي (Empirical Analysis)



شكل (10): فعالية خوارزمية سيب (Shell Sort Efficiency)

والجزء البرمجي، نأخذ بسطين متساويين شيا:

```
void ShellSort(int numbers[], int array_size)
{
    int i, j, increment, temp;
    increment = 3, // increment = Gap
    while (increment > 0)
    {
        for (j=0; j < array_size; j++)
        {
            i = j;
            temp = numbers[i];
            while ((i > increment) && (numbers[i - increment] > temp))
            {
                numbers[i] = numbers[i - increment];
                i = i - increment;
            }
            numbers[i] = temp;
        }
        if (increment > 1)
            increment = increment / 2;
    }
}
```

```

else if (increment == 1)
    increment = 0;
else
    increment = 1;
}
}

```

٥. خوارزمية الترتيب السريع (Quick Sort Algorithm)

الترتيب السريع هو طريقة ترتيب من اختراع هور (C A R Hoare) في 1962

خصائص خوارزمية

تتخذ الخوارزمية في عملها على وضع العنصر الأول (يسمى المؤشر) في مكانه النهائي ثم وضع العناصر الأكبر من المؤشر من جهة اليمين و العناصر الأصغر من جهة اليسار، وتسمى هذه العملية بمرحلة، ثم نقوم بحذف هذه المرحلة بمرحلة مقسمة لكل جهة (اليمين، اليسار)، حيث نحدد مؤشر جديدا ونعيد عملية التجربة متكررة هذه العملية إلى أن نحصل على هجوعه مرتبة

إذا لم نحدد المؤشر بطريقة صحيحة، نحصل على الطريقة الأسرع للترتيب في لحظة المعلومات مع تعقيد $O(n \ln n)$ والتي قد تتحول إلى $O(n^2)$ في الحالة الاصعب، و في حالة جنوب عناصره مرتبة أصلا، و لكن هذه الحالة بسيطة لأن المجموعة مرتبة أصلا

من الناحية العملية، بالنسبة للتجربة مع عدد قليل لا يحتاج وضع عشرات من العناصر، يتم التجزء عند إلى الترتيب و ثم ج يدي يكون أفضل من الترتيب السريع

و بصورة عامة يعتبر الترتيب السريع الأكثر شيوعا (مبسطة) من بين جميع خوارزميات الترتيب حيث المشكلة الوحيدة تحدث في حالة اختيار المؤشر.

اختيار أفضل مؤشر

عند استعمال الترتيب السريع مجموعة مرتبة مسبقا، و طريقة عملها، يستغرق وقتا وقت كبير، و ذلك بسبب أن نزل عنصر هو الذي يعتبر هو من الشيء الذي يؤدي إلى عدم استخدام المجموعة التي هي أكبر و أصغر من المؤشر. لحل المشكلة يتم اختيار عنصر الوسط، كما يمكن اختياره عشوائيا من عنصرين متتاليين حول المركز

تكون فكرة خوارزمية الترتيب باستخدام مبدأ التجربة حيث نقوم بعمل الخطوات التالية:

- 1- تقسم القائمة إلى جزئين حيث نأخذ عنصر للقسمه ولكن في الوسط تقريبا (يسمى Q)
- 2- نقوم بعملية الفصح بالجدولين بحيث تكون العناصر على جهة اليسار هي الأصغر من Q و إنا نقوم بتقسيل، و العناصر الموجودة على جهة اليمين فهي الأكبر من قيمة Q

3. نأخذ النصف الأول ونجري عليه عملية مشابهة سابقا سريع مرة أخرى كذلك نأخذ النصف الثاني وهكذا إلى أن تكون جميع العناصر مرتبة

(النصف الثاني (اليمين والأكبر) (X) (نصف النصف الأول (اليسار و الأصغر)

ملاحظة

* إذا كانت قيمة (N) هي عدد زوجي فإن قيمة (X) تكون

مثال N=8

$$8 \div 2 = 4$$

نكتبه كالمسطر : 1 2 3 4 5 6 7 8

* إذا كانت قيمة (N) هي عدد فردي فإن قيمة (X) تكون

مثال N=11

$$11 \div 2 = 5.5$$

5 or 6

نكتبه كالمسطر : 1 2 3 4 5 6 7 8 9 10 11

مثال 4. رتب العناصر التالية ترتيب تصاعدي باستخدام الترتيب السريع (Quick Sort Algorithm)

20 , 85 , 60 , 75 , 70 , 88 , 50 , 90 , 33 , 95

الحل // نوزع باستخدام الصغير من التالي

$$X = 10/2 = 5$$

المعرف X : العنصر الموجود في وسط القائمة
القيمة X = 70

F = Front : متعة القائمة ونكتب بالحد ()

L = Last : المؤخرة القائمة ونكتب بالحد ()

المرحلة الأولى X = 5

20 , 85 , 60 , 75 , 70 , 88 , 50 , 90 , 33 , 95

$$I = 1 \quad F = 1$$

$$J = 10 \quad L = 9$$

20 < 95 أي لا يوجد تبديل

20 85 60 75 70 88 50 90 33 95

$$I = 2 \quad F = 2$$

$$J = 9 \quad L = 9$$

85 < 33 أي يوجد تبديل

20 33 60 75 70 88 50 90 85 95

$I=3 \quad F=3$
 $I=8 \quad L=8$
 $60 < 90$ أي لا يوجد تبادل
 $20, 33, 60, 75, 70, 88, 90, 85, 95$

$I=5 \quad F=5$
 $I=6 \quad L=6$
 $70 < 88$ أي لا يوجد تبادل

$20, 33, 60, 90, 70, 88, 75, 90, 85, 95$

$I=4 \quad F=4$
 $I=7 \quad L=7$
 $50 < 75$ أي يوجد تبادل

$20, 33, 60, 90, 70, 88, 75, 90, 85, 95$
 $I=5 \quad F=5$
 $20, 33, 60, 90, 70, 88, 75, 90, 85, 95$

المرحلة الثانية:-

$20, 33, 60, 50, 70, 88, 75, 90, 85, 95$
 $I=1 \quad J=4 \quad I=6 \quad J=10$
 $\uparrow \quad \quad \quad \uparrow \quad \quad \quad \uparrow \quad \quad \quad \uparrow$

$X = 33$
 $70, 33, 60, 50$
 $\leftarrow X \quad \leftarrow$

$X = 90$
 $88, 75, 90, 85, 95$
 $\leftarrow X \quad \leftarrow$

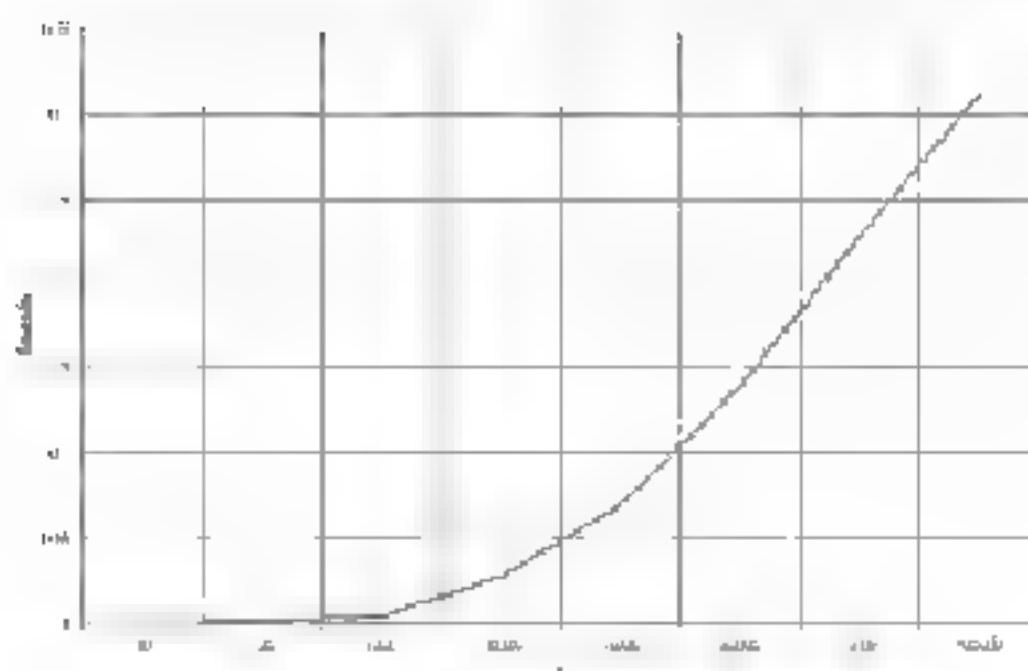
ملاحظة// في خوارزمية الفرز بالقرصعة المتفرقة الكثير من الوقت حاسبه ان نكن عدد
 العناصر كبير حيث ان
 $N \log_2 N$ عدد العمل
 $N \log_2 N^2$ عدد العمل للبيانات

مثال ١٠: رتب عناصر القائمة الآتية (١, ٢, ٣, ٤) باستخدام خوارزمية فرز سريع ^{*}

الحل // يمكن ذلك كما في الخطوات الآتية



تحليل تجريبي (Empirical Analysis)



شكل (11) فعالية خوارزمية فرز سريع (Quick Sort Efficiency)

والجزء اليماني الخاص بتطبيق خوارزمية التقسيم السريع هو

```
void quickSort(int numbers[], int array_size)
{
    q_sort(numbers, 0, array_size - 1)
}
void q_sort(int numbers[], int left, int right)
{
    int pivot, l_hold, r_hold;

    l_hold = left;
    r_hold = right;
    pivot = numbers[left];
    while (left < right)
    {
        while ((numbers[right] >= pivot) && (left < right))
            right--;
        if (left != right)
        {
            numbers[left] = numbers[right];
            left++;
        }
        while ((numbers[left] <= pivot) && (left < right))
            left++;
        if (left != right)
        {
            numbers[right] = numbers[left];
            right--;
        }
    }
    numbers[left] = pivot;
    pivot = left;
    left = l_hold;
    right = r_hold;
    if (left < pivot)
        q_sort(numbers, left, pivot - 1);
    if (right > pivot)
        q_sort(numbers, pivot + 1, right)
}
```

تجربتك الأولى

عند استعمال الترتيب السريع لترتيب هجوع عنه ثابت العناصر كثيره يمكن تغيير تعديه التي تلك عدد الوصول الى عنصره غير مرتبه عدد عناصرها صغير في 10 عناصر أي أن يكون الترتيب بالاختيار مناسب في هذه الحالة

مثال / استخدام برآل تقوم بتقسيم شفهه كثيره إلى هجوعين جريته الأصغر وشرطيها باستخدام حرايينه للترتيب السريع

```
typedef int tab_entiers[MAX],

int rapideEtape(tab_entiers t, int min, int max) {
    int temp = t[max];
    while(max > min) {
        while(max > min && t[min] <= temp) min++;
        if(max > min) {
            t[max] = t[min];
            max--;
            while(max > min && t[max] >= temp) max--;
            if(max > min) {
                t[min] = t[max];
                min++;
            }
        }
    }
    t[max] = temp;
    return max;
}

void rapide(tab_entiers t, int deb, int fin) {
    int mil;
    if(deb < fin) {
        mil = rapideEtape(t, deb, fin);
        if(mil > deb && mil < fin) {
            rapide(t, mil+1, fin);
            rapide(t, deb, mil-1);
        }
        else {
            rapide(t, deb, mil-1);
            rapide(t, mil+1, fin);
        }
    }
}
```

مثلاً برنامج يوضح كيفية امتداد النوار التي تقوم بعملية التقسيم السريع
(Quick Sort Algorithm)

```
Sort(A)
  Quicksort(A,1,n)

Quicksort(A, low, high)
  if ( low < high)
    pivot location = Partition(A,low,high)
    Quicksort(A,low, pivot location - 1)
    Quicksort(A, pivot location+1, high)

Partition(A, low, high)
  pivot = A[low]
  leftwall = low
  for i = low+1 to high
    if (A[i] < pivot) then
      leftwall = leftwall + 1
      swap A[i] A[leftwall]
  swap(A[low] A[leftwall])
```

الجدول التالي يوضح وقت التنفيذ لثلاثة خوارزميات (المتوسط، أفضل، وأقرب السريع) :-

method	statement	average time	worst-case time
insertion sort	9	$O(n^2)$	$O(n^2)$
shell sort	17	$O(n^{1.25})$	$O(n^{2.5})$
quicksort	21	$O(n \log n)$	$O(n^2)$

count	insertion	shell	quicksort
16	39 μ s	46 μ s	51 μ s
256	4,968 μ s	1,230 μ s	911 μ s
4,096	1.315 sec	.033 sec	.000 sec
65,536	436.427 sec	1.254 sec	461 sec

6. خوارزمية الترتيب الأسفل (ترقيعي) (Radix sort Algorithm)

سرع من الترتيب يعتمد على السرعة الموجودة في الارقام وتنقسم إلى حلقاء بحيث ترتب العناصر حسب الارقام (Pockets) على هذه الطريقة يستخدم ما يسمى الخلفاء (Digit) (0-9) بعدد الحلقاء في كل مرحلة بحيث تكون عدد الارقام مساوي في أكبر عدد الارقام في أكبر رقم.

مثال تطبيق // إذا كان لدينا العنصر التاليه (9 7 132)، فإن أكبر رقم يحتوي على 3 مراتب هو (132)

ملاحظة: من هذه الطريقة نحصل على عملية وذلك لإعطاء استخدام الاختيار في كل مرة بحيث نحصل (link Queue)، أي أن كل حلقه هي بمثابة طابور عنصر FIFO

مثال: هناك العناصر التالية وبها نستخدم لترتيب ترقيعي (Radix sort Algorithm).

42	23	74	11	65	58	94	36	99	87
Ordinal									

الحل: يمكن توضيحه بجدول لكل حلقه (مرحلة) وبك يعني:

0	1	2	3	4	5	6	7	8	9
	11	42	23	74, 94	65	36	87	58	99

1-Pockets 11, 42, 23, 74, 94, 65, 36, 87, 58, 99

0	1	2	3	4	5	6	7	8	9
	11	23	36	42	58	65	74	87	94, 99

2-Pockets 11, 23, 36, 42, 58, 65, 74, 87, 94, 99

7 ترتيب المؤشرات (Sorting Pointers Algorithm)

يستخدم هذه الطريقة لربط وترتيب العناصر حسب المؤشرات حيث يستخدم فكرة التبادل كما هي الحال التالي .

مثلاً: نأخذ العناصر الآتية بطريقة ترتيب المؤشرات (Sorting Pointers Algorithm)

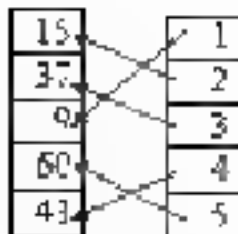
15 , 37 , 9 60 , 43

الخطوة: يمكن ترتيبها بخطوات الآتية .

- 1 - صنع العناصر لي خائف
- 2 - صنع المؤشرات

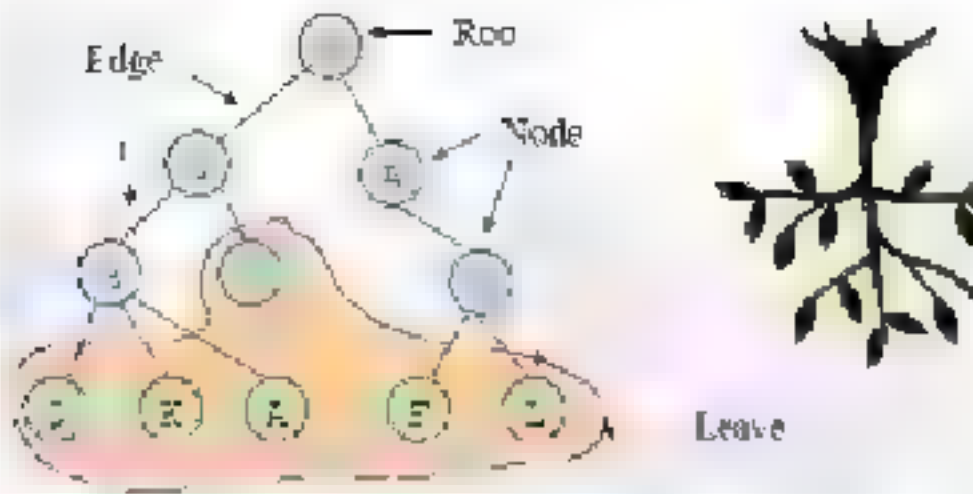


ب ترتيب المؤشرات



8 الترتيب بشجري شجرة بحث الثنائية (Tree for Binary search tree)

الشجرة هي بنية تكون فيها القيمة ثنائية لأي قيمة أكبر من البنية الأيسر به وبالعكس من قيمة البنية الأيمن لها. يسمف من الجذر (Root) والعقد (Nodes) والأوراق (Leaves) وهي خطوات معقدة وكذلك حذف الخطوات معقدة أيضاً ، والشكل رقم (12) الآتي يوضح الهيكل الشجري.



شكل رقم (2) الترتيب الشجري

مثال تطبيق: شجرة بحث ثنائية، يتم بناءها من حساب الأعداد التالية الشجرى



مثال: كود شجرة بحث ثنائية (Tree Binary Search) العناصر التالية بعد كل خطوة ؟
5, 9, 7, 3, 8, 12, 6, 4, 20

الحل: يمكن ذلك من خلال الخطوات الآتية:

1-

نحدد العنصر الأول فهو 5

2-

نحدد العنصر الثاني فهو 9، جوع 9 من 5 لأنه أكبر من 5

3-

(9)

3. نأخذ العنصر الثالث فنكون فرعاً من العنصر الثاني



4. نأخذ العنصر الرابع فنكون فرعاً من العنصر



5. نأخذ العنصر الخامس فنكون فرعاً من (7) (العنصر الثالث)



6. نأخذ العنصر السادس فنكون فرعاً من (9) (العنصر الثاني)



7 - نأخذ العنصر السابع (6) فنكون فرعاً من (7) (العنصر الثالث)



8. العنصر الثامن (4) نكرر فرعاً من (3) (العنصر الرابع)



9. العنصر 20 يكون فرع يفرع منه (2).



3 4 6 7, 8 9 12 20

بما اننا نستخدم الحذف الخاصة بالاشجار الثنائية فإنه يمكن توضيحها بالطرق الثلاث لاسف.

1- حذف عقدة نهائية (ورقة) - (Leave Delete):

لكي نقوم بحذف عقدة نهائية فائداً يجب ان نغير العقدة وننوي أن نؤشر على عقده بعد



2. حذف عقدة لها من واحد (One Child Node Delete):

لكي نقوم بحذف الحذف هذه تطبق ما يلي:

1- جعل المؤشر في العقدة يشير إلى العقدة الأولى

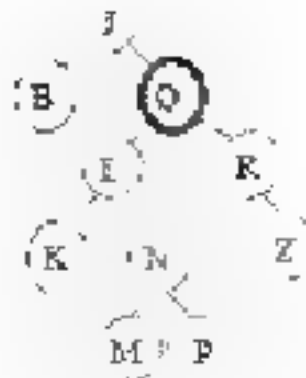
١-ا- نفي العقد المتصوب $dispose$



١-ب- حذف عقده بعد فرز (Two Childs Node Delete)

- ١- يتم ذلك من خلال الخطوات التالية
- ٢- استفس العقد المطلوب حذفه، بالعثور على العقد الذي له واحد من العقدتين B و C، ويتم استبدال عقدهما من الشجرة
- ٣- الفرعية اليسرى أو الشجرة الفرعية اليسرى للعقد المطلوب حذفه
- ٤- نسخة الشجرة الفرعية اليسرى للعقد الذي للعقد في السابق، العقد المطلوب حذفه، علف فئة B،
- ٥- يمكن لها فرع اليسر ليكون الفرع الأيمن جديد

الشجرة التالية توضح عملية حذف العقد Q واستبدالها بالعقد P





2 حذف الحقة {ق} وهي حقة بـالقائه (بوالقاء)



3 حذف النقطة (5) وهي عقد الحدي وتماسك حرفه نفس فقط. إذا كان الإتيان بها



4- حذف النقطه (6) : نقطه بهايه



5. حقائق حقة (7) . هذه هي نفس فقط لنا فهو شريك .



6. $\left(\frac{1}{2}\right)^n$ is the probability of getting n heads.



7- حذف عقدة (9) . تلك رقم ليس لها نود يربتها



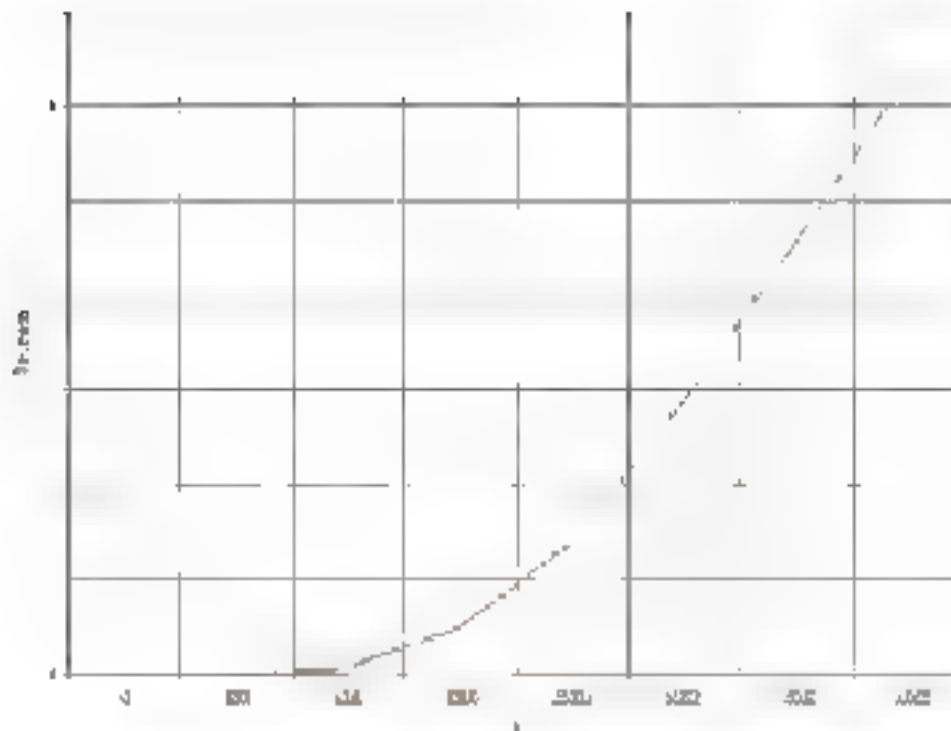
8- حذف عقدة (12) . تلك رقم ليس يربتها



وبوضوح هذه السلسلة يخزنها داخل المكدس كالتالي

3	4	5	6	7	8	9	10	11
---	---	---	---	---	---	---	----	----

نتحصل نتجوبي (Empirical Analysis).



شكل (13): فعالية خوارزمية الترتيب الشجري (Tree Sort Efficiency)

و جزء البرنامج الذي يقوم بعملية الترتيب للشجرة هو

```
void TreeSort(int numbers[], int array_size)
{
    int i, temp;
    for (i = (array_size / 2) - 1; i >= 0; i--)
        siftDown(numbers, i, array_size);
    for (i = array_size - 1; i >= 1; i--)
    {
        temp = numbers[0];
        numbers[0] = numbers[i];
        numbers[i] = temp;
        siftDown(numbers, 0, i - 1);
    }
}

void siftDown(int numbers[], int root, int bottom)
{
    int done, maxChild, temp;
    done = 0;
    while ((root * 2 <= bottom) && (!done))
    {
        if (root * 2 == bottom)
            maxChild = root * 2;
        else if (numbers[root * 2] > numbers[root * 2 + 1])
            maxChild = root * 2;
        else
            maxChild = root * 2 + 1;

        if (numbers[root] < numbers[maxChild])
        {
            temp = numbers[root];
            numbers[root] = numbers[maxChild];
            numbers[maxChild] = temp;
            root = maxChild;
        }
        else
            done = 1;
    }
}
```


٩. خوارزمية الترتيب التوبولوجي (Topological sorting Algorithm)

تستخدم خوارزمية الترتيب هذه فكرة للمصنفات ووضح التيم يانطير، ويمكن توضيحها بالمثل التالي



The graph shown to the left has many valid topological sorts, including

- 1,5,3,11,8,2,10,9
- 7,5,11,2,3,10,8,9
- 3,7,8,5,11,10,9,2
- 2,5,7,11,10,3,8,9

من المصنفات الطوبولوجية هي: وقت التنفيذ خطي، يعتمد بريفيد العقد بين المسارات المختلفة، وقد استخدمت إحدى الخوارزميات كانت من قبل العالم (Kahn 1962)، معتمدة لفكرة الطوبولوجيا

حيث: E

E = الحواف

Q = الطوبولوجيا

V = الرؤوس

والخوارزمية التي توضح الطريقة هي

```

L ← Empty list where we put the sorted elements
Q ← Set of all nodes with no incoming edges
while Q is non-empty do
  remove a node n from Q
  insert n into L
  for each node m with an edge e from n to m do
    remove edge e from the graph
    if m has no other incoming edges then
      insert m into Q
if graph has edges then
  output error message (graph has a cycle)
  
```

else

output message (proposed topologically sorted order 1)

2-4. في رسم الترتيب الحد هي (External Sorting Algorithms).

1. حوار بعد ترتيب التبعج (Merge sort Algorithm).

هذه، تقوم بترتيب مجموعة عناصر بطريقة التبعج كما في الجدول التالية

مرحلة	T1	T2	T3
A	7 8 8 4	0 9	
B	6		9 8 7 4
C	0 8 7 4	2 6	
D			9 8 7 6 4

وجاء الترتيب الخاص بتطبيق طريقة ترتيب التبعج هو

```
void MergeSort(int numbers[], int temp[], int array_size)
{
    m_sort(numbers, temp, 0, array_size - 1);
}
void m_sort(int numbers[], int temp[], int left, int right)
{
    int mid;
    if (right > left)
    {
        mid = (right + left) / 2;
        m_sort(numbers, temp, left, mid);
        m_sort(numbers, temp, mid+1, right);
        merge(numbers, temp, left, mid+1, right);
    }
}
```

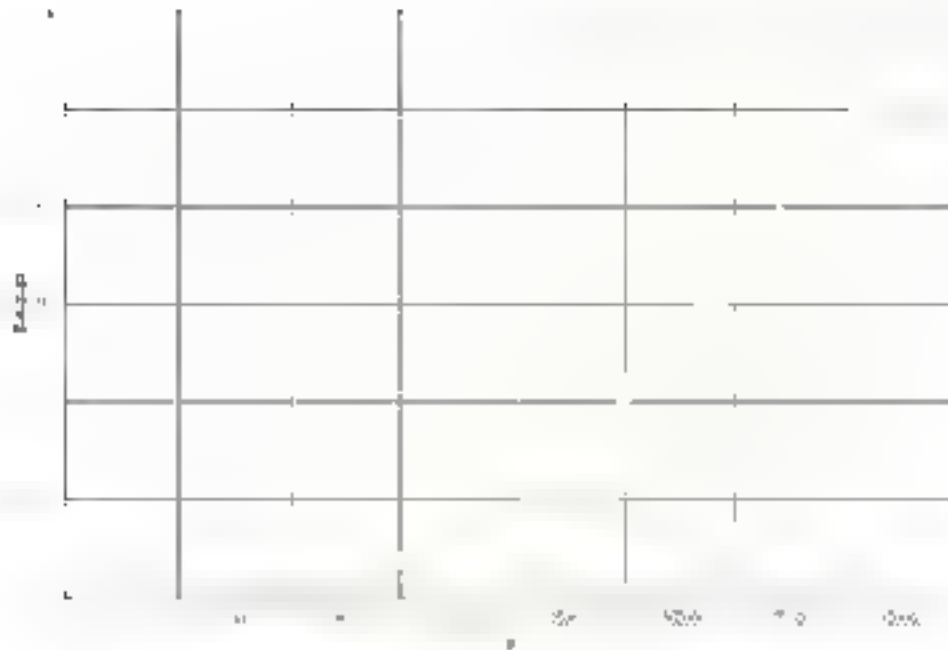
```

}
void merge(int numbers[], int temp[], int left, int mid, int right)
{
    int i, left_end, num_elements, tmp_pos;
    left_end = mid - 1;
    tmp_pos = left;
    num_elements = right - left + 1;

    while ((left <= left_end) && (mid <= right))
    {
        if (numbers[left] <= numbers[mid])
        {
            temp[tmp_pos] = numbers[left];
            tmp_pos = tmp_pos + 1;
            left = left + 1;
        }
        else
        {
            temp[tmp_pos] = numbers[mid];
            tmp_pos = tmp_pos + 1;
            mid = mid + 1;
        }
    }
    while (left <= left_end)
    {
        temp[tmp_pos] = numbers[left];
        left = left + 1;
        tmp_pos = tmp_pos + 1;
    }
    while (mid <= right)
    {
        temp[tmp_pos] = numbers[mid];
        mid = mid + 1;
        tmp_pos = tmp_pos + 1;
    }
    for (i=0; i <= num_elements; i++)
    {
        numbers[right] = temp[right];
        right = right - 1;
    }
}

```

التحليل التجريبي (Empirical Analysis)



شكل (14) كفاءة خوارزمية الدمج (Merge Sort Efficiency)

2- خوارزمية دمج الدمج المتوازنة (Balanced Two-way Merge Sort)

(Balanced Two-way Merge Sort)

- يمكن توصيف فكر هذه الطريقة بالمثل الأسى الخاص بخلقة شجرة كهذا في الصور التالية
- 1 تقسم القائمة إلى قسمين متساويين تقريباً وتسمى A و B. ويصبح كل عنصر من B مع نظيره الأول في القائمة A.
- 2 نقارن العنصر الأول في القائمة B مع العنصر نظيره الثاني في القائمة A ونصنفه في القائمة C بالترتيب.
- 3 نقارن العنصر الثاني في القائمة B مع العنصر نظيره الثاني في القائمة A ونصنفه في القائمة D بالترتيب.
- 4 نكرر الخطوات 2 و 3 لنصنف على عناصر نظيرها 2 في كل من القامتين A و B.
- 5 نصنف العنصر بالترتيب في القامتين A و B.
- 6 نضع الطريقة نقوم بدمج عناصر القامتين A و B حيث عناصره بوضوح 4 لتكون مرتبة ونصنفها في القائمة C و D.
- 7 نعيد الطريقة بدمج عناصر القامتين A و B بطول 8.
- 7 نستقر بهذا الطرف على الوصول إلى قائمة مرتبة

43 60 94 43 41 17, 11 5 ختمه من قبله

هذه الطريقة لها مساوئ هي

- 1- إنها تحتاج إلى مصفوفة خزن أمثلية يمكن أن تكون أكبر من المصفوفة الأصلية (n^2) على عدد العناصر n في n .
- 2- تحتاج إلى مصفوفات فرعية عندئذ يكون ، هذه المصفوفات بعض n في n مستخرج وهذا يعني أن الوقت المطلوب لإكمال عملية الترتيب كبير نسبة إلى غيرها من الطرق .

لا تستخرج من ذلك بأن

عدد المراحل $\log n$ No. Of Pass
عدد المقارنات $n \log n$ No. Of Comparison

الفصل الثالث

البحث

Searching

3.1 البحث (Search)

البحث هي عملية إيجاد عنصر معين في مجموعة من البيانات فإذا كان العنصر موجود في المجموعة العنصرية تفكير إيجابية وإذا لم يتكون سلبية في حالة عدم وجوده ، ولكن تكون العملية صالحة يحصل أن تكون العنصر مرتبة

3.2- البحث التسلسلي (Sequential Search)

في عملية البحث عن عنصر من خلال مسح أو استعراض عناصر القائمة من بدايتها وبالنسبة لحين الوصول لعنصر المطلوب إذا كان موجودا ما في حالة الوصول نهاية القائمة ولم يحصل عملية يعني أن العنصر غير موجود

عدد المقارنات: $n/2$

وقت التنفيذ: $O(n)$

مثال: البرنامج التالي يقوم بالبحث عن عنصر من بين مجموعة عناصر باستخدام البحث التسلسلي عفاً عن عدد العناصر (7) والعنصر المراد بالبحث عنه (3) والعنصر هو

data[] = 7,4,5,6,3,9,10

الحل://

```
#include<stdio.h>
main( )
{
    int data[ ]={7,4,5,6,3,9,10};
    int nmax=7;
    int key=3;
    printf("%d\n",sqsearch(data,nmax,key));
}
Sqsearch(data,n,k),
int data[ ]
int n,
int k,
{
    Register int i
    For (i=0;i<=n; ++i)
    If (k==data[ i ])return(i+1)
    Return(-1); /* no match exist */
}
```

نتيجة البحث هي أن العنصر (3) موجود في القائمة بالموقع (5)

3-3. البحث الثنائي (Binary search).

تقوم فكرة البحث الثنائي على تقسيم المصفوفة إلى نصفين واستبعاد النصف الذي لا يتلقى إليه المصباح key الذي يبحث عنه عن طريق تحديد العنصر الذي يقع في منتصف هذه المصفوفة، ثم يكرر هذا العنصر مع المفتاح الذي يبحث عنه (المصفوفة مرتبة ترتيباً) .
وال Pseudo code التالي يوضح لك هذه الطريقة.

```
repeat
If ID <= Array[k] then r=k-1
If ID >= Array[k] then l=k+1
until l>=r
If l==j then we found the ID in the array
else the ID is not found
```

مثال : مصفوفة مرتبة ترتيباً أليحي

```
word[] = {"begin", "const", "do", "end", "if", "odd", "program", "read",
"then", "var", "while", "write"}
```

كيف نكتب code محواري للبحث الثنائي ؟ كيف يبحث في المصفوفة بترتيب؟
نقوم بمقارنته بعنصر في منتصف عن طريق تقاطعها مع مفتاحه لتسهيل البحث في.

```
strcmp (char *str1, char *str2)
```

هذه الدالة تقوم بمقارنته حرفين أو مستطتين حرفيين ونقوم بإرجاع:

التيه (صفر) إذا كانت str1 = str2

قيمة سالبة إذا كانت str1 < str2

قيمة موجبة إذا كانت str1 > str2

و حرم البرنامج الذي نقوم بتطبيق خوارزمية البحث الثنائي (binary search) هي .

```
#include "STRING.H"
#include "STDIO.H"
#define max_size 12
//
int binary_search (ID)
char *ID
{
char *word[] = {"begin", "const", "do", "end", "if", "odd", "program",
"read", "then", "var", "while", "write"},
int i=0, j= max_size-1, s, k,
while(s<=j)
{
k=(i+j)/2;
s=strcmp(ID, word[k]);
if (s<=0) j=k-1,
if (s>=0) i=k+1,
}
```

```

if((i-1)>j){
    printf("have found the key (%s) at element %d", word[k] k+1);
    return k
}
return -1,
}
}
void main()
{
    int result;
    char *ID;
    printf("\nFiz. Enter the ID to begin search\nID=");
    scanf("%s",ID);
    result = binary_search(ID);
    if (result==1){
        printf("the key(%s) is not found" ID) }
    getch();
}

```

والتي تطبق خوارزمية البحث الثنائي (Binary Search) على مصفوفة م مبع الخسوف
المتخصصة التالية

- 1- الخطوة الأولى: وإذا لم بالنسبة لا يمكن تطبيق الخوارزمية إلا إذا كانت العناصر
مرتبة تصاعدياً أو تنازلياً أو فصحاً على حسب نوع العلاقات المتعددة فيها .
- 2- تحديد أول عنصر في المصفوفة الحرف 1 ، وآخر عنصر فيها والعرفاً مثلاً 7
- 3- تحديد العنصر الذي يقع في منتصف هذه المصفوفة الحرف 4
- 4- بعد ذلك يمكن تطبيق البحث الثنائي على مصفوفة إلى كين

أ- إذا كان يساويه يكون قد وجد عنصر الذي يبحث عنه
ب- إذا كانت قيمة المفتاح أقل من قيمة العنصر الأوسط في المصفوفة، إذن نحتاج أن نبحث
فقط في نصف المصفوفة الأول ونستبعد البحث في النصف الثاني
وهذا عند ذلك إذا كانت قيمة المفتاح أكبر من قيمة العنصر الأوسط في المصفوفة من نحتاج
أن نبحث فقط في نصف المصفوفة الثاني، ويستبعد البحث في النصف الأول

حيث نعتبر النصف الذي حسبنا البحث فيه مصفوفة قائمة بحددها، نحدد فيها $i, j, & k$
(أي نقوم بتقسيمها إلى قسمين) ونطبق نفس الخطوات من 1 إلى 3 فيها، ثم نقرر الفتح مع
العنصر الأوسط الحدد نفس التوقيت الذي نكرر في الخطوة 1 إلى 3 السابقة.

أن خوارزمية هذا البحث تقوم بالبحث عن عنصر في قائمة مرتبة

- 1- تحديد موقع العنصر الذي يقع في منتصف القائمة تقريباً
- 2- إذا كان العنصر المطلوب (Item) مساوياً لعنصر في الوسط (X) فطبي ذلك أنه عملية البحث
مع العنصر الذي في الوسط انتهت، إذا كانت أقل من قيمة العنصر في الوسط نستبعد
البحث في النصف اليسرى و أقل من قيمة العنصر (X) ، أما إذا كان العنصر الذي سبحث عنه
أكبر من قيمة (X) فنكون البحث في النصف اليمى ولا نكرر

3- في الحالتين، علاوة على تتم المعالجة بنفس الطريقة التي سوف يجرى التفاوض للمعقود بحسب
الوصول إلى الخصم المطلوب أي في

٥٥٥ الطاهر بن محمد

عدد المخطوطات: 1094م.



هذا هو المقصود أننا نحط على عناصر المنطقة في هذه الصورة

Алгебра: 0, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28,

الحمد لله رب العالمين

القول: لا يمكن توضيحه بطريقين: الأولى

```
#include "STRING.h"
#include "STDIO.h"
#define max_size 15

//=====

int binary_search (key)
int key
{
int NumArray[]={0, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28};
int i=0, j=max_size-1, k=(i+j)/2,
while(i<=j)
    if (key == NumArray[k]) ,
        printf("we found the key (%d)", key)
        return k,
    }
else{
    if (key < NumArray[k]){
        j = k,
        k=(i+j)/2,
    }
    if (key > NumArray[k]){
        i = k,
        k=(i+j)/2,
    }
}
}
return -1
```

```

}
}

void main()

int result;
int key;
printf("\nPlz. Enter the Key to begin search:\n");
scanf("%d",&key);
result = binary_search(key);
if(result== -1){
printf("the key(%d) is not found",key); }
getch();
}

```

عند م اء البحث في "ي مصفوفة عن عنصر محدد باستخدام بحث (Binary Search)

ان أقصى عدد من مر ثء البحث باستخدام Binary Search في ي مصفوفة يعطى من بحلا القوة التي مخرج اليا رقم (2) مكي نصف العدد متى يريد عن عنصر المصفوفة هو حد، أي أنه لو قوة 1 (2) واليا تُعطى رقم أكبر من حد عناصر المصفوفة يربط

في مثال استخدم مصفوفة من (15) عنصر، ملاحظ أن بعد قمتي يريد على عدد عناصر المصفوفة هو حد، أي العدد (16) ناتج من القوة التي مخرج رقم (2) هي (16=4*2)، مكي يعني هذا احتياج على أكثر أربع مرات مرقته في ال Binary Search حتى بعد العنصر الذي يبحث عنه، من الممكن ان نجد من أول مرة في المخرجة أو نجد في ثاني مرة، أو ثالث مرة أو أربع مرة أو لم يكن غير موجود في المصفوفة

عدد المقارنات هي $\log_2 n$ مثال: مكي القيمة العليا المطلوب البحث عن العنصر $key=29$ باستخدام طريقة البحث الثنائي (Binary Search)

9	11	16	18	25	29	32	35
---	----	----	----	----	----	----	----

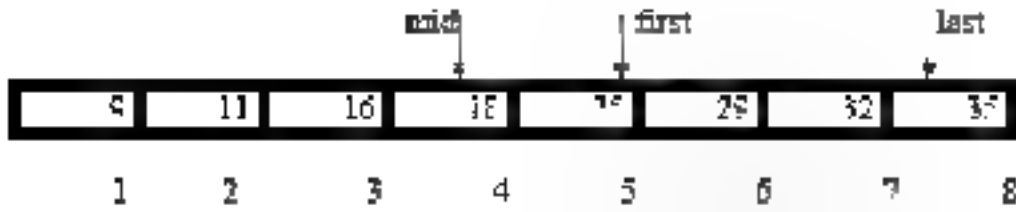
الحل: 1
 $I=1=first$
 $J=8=last$
 $Key=29$

1							
9	11	16	18	25	29	32	35
1	2	3	4	5	6	7	8

2- نجد القيمة الوسطية $mid = (1+8) \div 2 = 4$

$List[4] = 25$

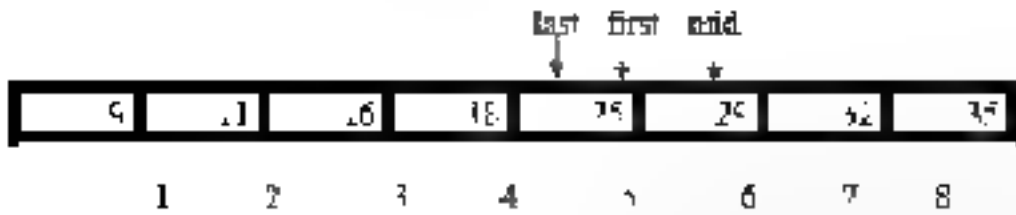
3 من يكرر $first = mid + 1$



4 نجد $mid = (5+8) \div 2 = 6$

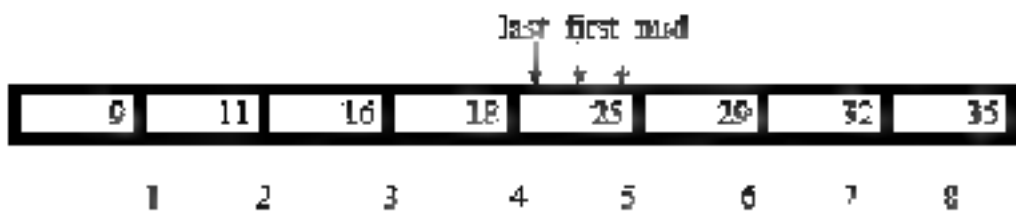
$List[6] = 26$

$Last = mid - 1$



5 نجد $mid = (5+5) \div 2 = 5$

$List[5] = 25$



النتيجة العنصر موجود بالموقع 5

مثال: استعمل الجدول التالي عن العناصر التالية بطريقة البحث الثنائي

Data[]=15,6,0,7 9,23,54 82,101

البحث / للعناصر المراد البحث عنها هي

X=101 x=14 x=82

X=101	Low=i=first	High=j=last	m.d
	1	9	5
	6	9	7
	8	9	8
	9	9	9

$m.d = (low + high) \div 2$

Found=101 في الموضع 9

X=14	Low=i=first	High=j=last	m.d
	1	9	5
	1	4	?
	1	1	1
	2	1	Not found

وسط التوقف هنا $low > high$

X=82	Low=i=first	High=j=last	m.d
	1	9	5
	6	9	7
	8	9	8

العناصر 82 موجودة في الموضع 8

يمكن إيجاد معدل عدد المقارنات لكلية عناصر باستخدام القانون التالي.

Average of comparison=sum of comparisons÷number of elements

	1	2	3	4	5	6	7	8	9
element	15	6	0	7	9	23	54	82	101
comparisons	3	2	3	4	1	3	2	3	4

Average of comparison=25/9=2.77

4- البحث في الشجرة ثنائية (Binary Tree Search):

شجرة البحث الثنائية هي الشجرة التي كل عقدة فيها لكن من عقدها في اليسار منها، وأقل من عقدها الذي في اليمين منها، الشكل التالي يوضح ذلك.



شكل (S) : شجرة بحث ثنائية

تعد استخدام الشجرة الثنائية للبحث قائمة من الأعداد، يمكن البحث عن عدد ما، حيث أنه بحث المقدم التي بحرفه ونظف، وفق ما يلي.

إذا كتب المقدم المراد حذفه ورقاه، يمكن حذفه بون تغيير بحر في الشجرة.

إذا لم يكن العقدة التي لا حلقه ورقاه يجب يتم حذفها وتتمتع مكانه المقدم التي بحرفي العدد التالي وفقاً للترتيب المتصاعدي للأعداد المخرجة في الشجرة.

وجزاء اليركصح الخاص بتطبيق البحث في الشجرة السابقة هو

```

#include<iostream.h>
struct nodetype
{
    int k;
    struct nodetype* left;
    struct nodetype* right;
};
typedef struct nodetype* nodeptr;
nodeptr maketree(int x)
{
    nodeptr p;
    p=new nodetype;
    p->k=x;
    p->left=NULL;
    p->right=NULL;
    return (p);
}
  
```

```

void build(nodeptr node, int number)
{
    if(number==node->k)
        if(node->right==NULL)
            node->right=makeTree(number);
        else
            build(node->right,number);
    else
    {
        if(number<node->k)
            if(node->left==NULL)
                node->left=makeTree(number);
            else
                build(node->left,number);
        else
            cout<<"Duplicate number "<<number<<endl;
    }
    return;
}

void search(int key, nodeptr root)
{
    nodeptr p, f, q, rp, s;
    p=root, // p will point to the node
    q=NULL, // and q to its father, if any.
    while(p!=NULL && p->k!=key)
    {
        q=p
        if(key<p->k)
            p=p->left;
        else
            p=p->right;
    }
    if(p==NULL)
        cout<<"The key does not exist in the tree\n" //leave the tree
        unchanged;
    else // rp will point to the node that will replace node p
        if(p->left==NULL) // node p has right son only
            rp=p->right;
        else
            if(p->right==NULL) // node p has left son only
                rp=p->left;
            else // node p has two sons

```



```

{
    f=p;
    rp=p->right;
    s=rp->left
    while(s!=NULL)
    {
        f=rp
        rp=s;
        s=rp->left; // s is always the left son of rp
    } // now rp is the inorder successor of p
    if(f!=p) // if p is not the father of rp
    {
        f->left=rp->right;
        rp->right=p->right;
    }
    rp->left=p->left;
}
if(q==NULL) // if p was the root of the tree
    root=rp;
else
    if(p==q->left)
        q->left=rp;
    else
        q->right=rp;
delete(p);
}
void print(nodeptr p)
{
    if(p!=NULL)
    {
        print(p->left);
        cout<<"p->key:"<<" ";
        print(p->right);
    }
    else
        cout<<" ";
}
}

```

```

void main()
{
    nodeptr tree;
    int number;
    cin >> number;
    tree = make_tree(number);
    while (cin >> number, number != 0)
        balok(tree, number);
    print(tree);
    cout << "Enter the number you want search for and delete" << endl;
    int n;
    cin >> n;
    search(n, tree);
    print(tree);
}

```

اما جزء الخوارزمية الخاص بإصلاحه فقد انشجرة البحث اثنائه فيكون كالآتي:

```

T cc insert(x)
    u ← NIL
    z ← new x;
    while (u ≠ NIL)
        do
            y ← u
            if key[x] < key[y]
                then z ← left[y]
            else z ← right[y]
        if u ← z
    if y ← NIL
        then root[T] ← z
    else if key[z] < key[y]
        then left[y] ← z
    else right[y] ← z

```

y is maintained as the parent of x, since x eventually becomes NIL

The final test establishes whether the NIL was a left or right turn from y.

3. 5- بهید- حواریه لیج- (Complex Of Algorithm Search)

- [illegible]

$$T(n) = O(n \log n)$$

وَلَكِنْ لَقَدْ أَلْهَمَ الْأَعْيُنَ بِحَاسِبِهِ فَقَرَأَ فِيهِ بِحَسْبِ الْإِسْمِ

1- العمل في المؤسسة (Basic Operation)

بعد دراسة عقد جامعة مركز على الطلبة المؤسسة هذه الخويرة مثلت في
حريته مثلت الحق الطاعة المؤسسة في طاعة الخيرية على الطاعة التي في الحد هم رقم
محرم عة للتحسين كلمة كم عدد عميل فاعلية في كلف الحويرة كلف فاعلية

هذه هي الأخطار المحدقة بالبحر الأبيض المتوسط

```
int sum = 0;
for ( int i = 1; i <= n; i++ )
    sum += i;
```

نلاحظ أن كل من المتغيرات $i = 1, \dots, n$ و $j = 1, \dots, n$ هي متغيرات عشوائية.

تم التعميمات ($n \rightarrow n+1$) مرة بمرات عديدة مع تحقيق هذه الخصائص.

$$f(\underline{u}) = \underline{z} \quad ?$$

يقولون عن هذه الخوارزمية أنها ذات تعقيد خطي (Linear Complexity) لأننا نحتاج فقط

2. النقيض المتطارب (Asymptotic Contradiction).

بعد إيجادنا في بعض جوانب رعيه وروايتي قتلنا عندنا لافطيات واكثير اهل القصر الصغير ووبعد قتلنا بعد الاثني عشر يوم من القتل في القصر الصغير بعد الاثني عشر يوم من القتل في القصر الصغير

$$f(x) = x^3 + 5x^2 + a - 10$$

يتميز المؤلف 7 كتب هي: 1. 2. 3. 4. 5. 6. 7. 8. 9. 10. 11. 12. 13. 14. 15. 16. 17. 18. 19. 20. 21. 22. 23. 24. 25. 26. 27. 28. 29. 30. 31. 32. 33. 34. 35. 36. 37. 38. 39. 40. 41. 42. 43. 44. 45. 46. 47. 48. 49. 50. 51. 52. 53. 54. 55. 56. 57. 58. 59. 60. 61. 62. 63. 64. 65. 66. 67. 68. 69. 70. 71. 72. 73. 74. 75. 76. 77. 78. 79. 80. 81. 82. 83. 84. 85. 86. 87. 88. 89. 90. 91. 92. 93. 94. 95. 96. 97. 98. 99. 100. 101. 102. 103. 104. 105. 106. 107. 108. 109. 110. 111. 112. 113. 114. 115. 116. 117. 118. 119. 120. 121. 122. 123. 124. 125. 126. 127. 128. 129. 130. 131. 132. 133. 134. 135. 136. 137. 138. 139. 140. 141. 142. 143. 144. 145. 146. 147. 148. 149. 150. 151. 152. 153. 154. 155. 156. 157. 158. 159. 160. 161. 162. 163. 164. 165. 166. 167. 168. 169. 170. 171. 172. 173. 174. 175. 176. 177. 178. 179. 180. 181. 182. 183. 184. 185. 186. 187. 188. 189. 190. 191. 192. 193. 194. 195. 196. 197. 198. 199. 200. 201. 202. 203. 204. 205. 206. 207. 208. 209. 210. 211. 212. 213. 214. 215. 216. 217. 218. 219. 220. 221. 222. 223. 224. 225. 226. 227. 228. 229. 230. 231. 232. 233. 234. 235. 236. 237. 238. 239. 240. 241. 242. 243. 244. 245. 246. 247. 248. 249. 250. 251. 252. 253. 254. 255. 256. 257. 258. 259. 260. 261. 262. 263. 264. 265. 266. 267. 268. 269. 270. 271. 272. 273. 274. 275. 276. 277. 278. 279. 280. 281. 282. 283. 284. 285. 286. 287. 288. 289. 290. 291. 292. 293. 294. 295. 296. 297. 298. 299. 300. 301. 302. 303. 304. 305. 306. 307. 308. 309. 310. 311. 312. 313. 314. 315. 316. 317. 318. 319. 320. 321. 322. 323. 324. 325. 326. 327. 328. 329. 330. 331. 332. 333. 334. 335. 336. 337. 338. 339. 340. 341. 342. 343. 344. 345. 346. 347. 348. 349. 350. 351. 352. 353. 354. 355. 356. 357. 358. 359. 360. 361. 362. 363. 364. 365. 366. 367. 368. 369. 370. 371. 372. 373. 374. 375. 376. 377. 378. 379. 380. 381. 382. 383. 384. 385. 386. 387. 388. 389. 390. 391. 392. 393. 394. 395. 396. 397. 398. 399. 400. 401. 402. 403. 404. 405. 406. 407. 408. 409. 410. 411. 412. 413. 414. 415. 416. 417. 418. 419. 420. 421. 422. 423. 424. 425. 426. 427. 428. 429. 430. 431. 432. 433. 434. 435. 436. 437. 438. 439. 440. 441. 442. 443. 444. 445. 446. 447. 448. 449. 450. 451. 452. 453. 454. 455. 456. 457. 458. 459. 460. 461. 462. 463. 464. 465. 466. 467. 468. 469. 470. 471. 472. 473. 474. 475. 476. 477. 478. 479. 480. 481. 482. 483. 484. 485. 486. 487. 488. 489. 490. 491. 492. 493. 494. 495. 496. 497. 498. 499. 500. 501. 502. 503. 504. 505. 506. 507. 508. 509. 510. 511. 512. 513. 514. 515. 516. 517. 518. 519. 520. 521. 522. 523. 524. 525. 526. 527. 528. 529. 530. 531. 532. 533. 534. 535. 536. 537. 538. 539. 540. 541. 542. 543. 544. 545. 546. 547. 548. 549. 550. 551. 552. 553. 554. 555. 556. 557. 558. 559. 560. 561. 562. 563. 564. 565. 566. 567. 568. 569. 570. 571. 572. 573. 574. 575. 576. 577. 578. 579. 580. 581. 582. 583. 584. 585. 586. 587. 588. 589. 590. 591. 592. 593. 594. 595. 596. 597. 598. 599. 600. 601. 602. 603. 604. 605. 606. 607. 608. 609. 610. 611. 612. 613. 614. 615. 616. 617. 618. 619. 620. 621. 622. 623. 624. 625. 626. 627. 628. 629. 630. 631. 632. 633. 634. 635. 636. 637. 638. 639. 640. 641. 642. 643. 644. 645. 646. 647. 648. 649. 650. 651. 652. 653. 654. 655. 656. 657. 658. 659. 660. 661. 662. 663. 664. 665. 666. 667. 668. 669. 670. 671. 672. 673. 674. 675. 676. 677. 678. 679. 680. 681. 682. 683. 684. 685. 686. 687. 688. 689. 690. 691. 692. 693. 694. 695. 696. 697. 698. 699. 700. 701. 702. 703. 704. 705. 706. 707. 708. 709. 710. 711. 712. 713. 714. 715. 716. 717. 718. 719. 720. 721. 722. 723. 724. 725. 726. 727. 728. 729. 730. 731. 732. 733. 734. 735. 736. 737. 738. 739. 740. 741. 742. 743. 744. 745. 746. 747. 748. 749. 750. 751. 752. 753. 754. 755. 756. 757. 758. 759. 760. 761. 762. 763. 764. 765. 766. 767. 768. 769. 770. 771. 772. 773. 774. 775. 776. 777. 778. 779. 780. 781. 782. 783. 784. 785. 786. 787. 788. 789. 790. 791. 792. 793. 794. 795. 796. 797. 798. 799. 800. 801. 802. 803. 804. 805. 806. 807. 808. 809. 810. 811. 812. 813. 814. 815. 816. 817. 818. 819. 820. 821. 822. 823. 824. 825. 826. 827. 828. 829. 830. 831. 832. 833. 834. 835. 836. 837. 838. 83

$$f(x) = x^3$$

3- التحقق الرسمي في الحالات الأفضل والأسوأ والمتوسطة لخوارزمية (Beast & Worst & Average Complexity)

في المثال السابق دمج يتحقق في نفسه تلقائياً ولكن في معظم المصفوفات عند الانتهاء / مرور
فقط بقيمة الحجم (n) ليس يتعلق أيضاً بالمتوسط المتكافئ المتكافئ مستخدم حقه للبحث
الاحتمالي عن العنصر في نفسه من العنصر حيث أنه في هذه الحالة أربعة مقترنة العنصر x الذي
يبحث عنه مستخدم في array المخزنة في مصفوفة d وطول المصفوفة d = n وإرجاع لكل
العنصر في حل العثور عليه أو إرجاع القيمة -1 في حل عدم العثور عليه

مثال / جزء من الجي ليصبح تحقيقاً بسيطاً عن العنصر x في قائمة مصفوفة d[]

```
int j = 0;
while (j < n && d[j] != x)
    j++;
if (j < n)
    return j;
return -1;
```

مثال / برنامج يحدد مفهوم التحقق Complexity على أن أفضل الأسلوب على الترتيب في البرنامج
(قد لا تكون جميع هي x دقة 1)

```
{
    int mid;
    int first = 0;
    int last = N - 1;
    while (first <= last)
    {
        mid = (first + last) / 2;
        if (list[mid] == target)
            return mid;
        if (list[mid] > target)
            last = mid - 1;
        else first = mid + 1;
    }
}
```

في خطوات الحلقة الأفضل مصفوفة على هذه تكون العنصر المطلوب إيجاده هو نفسه
العنصر المتصل به وبالتالي لنحتاج لتفكيكه مقارنه وحيد

أما التعيينات في البحث الأمثل إليها قد يحتاج لتعقيد رياضي لإيجادها فلو كان لدينا مجموعة من الأسماء موزونة ضمن معطى n كان في الشكل، والى البحث على اسم "جورج" وقد وجدنا سجله في ثابت خطه n باستطاعة حوّل رتبة البحث الثاني، نصلح ستكون بحاجة إلى 7 خطوات لإيجاده باستخدام البحث الخطي.

begin			mid			end		
0	1	2	3	4	5	6	7	
[محمد	علاء	مكي	موسى	أحمد	فوزي	عبد	حسن
			begin			mid		end
					begin	mid		end
						begin	end	

ويمكن بعد العرف التي تخص هذه الخطوة `while` متالياً مع عدد العرف التي يجب علينا فيها تغيير المسألة n الحجم n على 2 وذلك حتى الوصول لمسألة فرعية في عنصر واحد. وتنتهي بهذا العنصر، وهو العنصر المطلوب.

في الحالة n غير زوجية من الأسفل 2 (نوعين متساويين) طالما أننا نبدأ بحجم المسألة الناتجة على 2 عنصر على الأقل (مستوى) جديد. وبالتالي في التحقق في هو التحول هو n مستقر في القسم على 2 حتى نصلح العنصر المطلوب في آخر خطوة القسم هو n نصفي بدلاً من التالي مستطاع بين $n/2$ حصراً كسوة بعدد وكلي متساوي سيكون 3 خطوات مسح وجوهر 8 عناصر.

أما خطوة التقيد الوسطى فانه لا يمكن الوصول لمسألة نهائية فمثلاً وجود العنصر في مكانه في المسألة هو احتمال وجود العنصر في المسألة الأصلية ضرورياً بحدوث وجود في المسألة الفرعية الثانية ضرورياً بحدوث وجود في المسألة الفرعية التي بعدها وهكذا. بالتالي نجد في البحث الثاني عدد يكثير من البحث الخطي وذلك لاقتصره عدد كبير جداً من العناصر في حال كانت n كبيرة بشكل كبير. مثلاً عندما $n=100000$ فإن أوقات البحث تحصل عليه في حالة البحث الخطي هو $n=100000$.

أما في حال البحث الثاني فإن أوقات البحث سيكون $\log_2(100000)=16$ أي من البحث الثاني وفر لنا 999984 عملية مقارنة نسبة للبحث الخطي.

الفصل الرابع
الامتثالية في مسائل تصميم
الخوارزميات

**Optimization in)
Algorithms Design
(Equations**

1.4 المخطط (Graph)

المخطط عبارة عن مجموعة من العناصر التي تمثل نقاط وروابط (vertices)
و حيث العناصر التي تربط بعلاقات تسمى حواف (Edges) وهذه العلاقات تمثل بحدود كما في شكل رقم (1.6) التالي



شكل رقم (1.6) يوضح مخطط غير متجه بسيط

$$G = (V, E)$$

$$V(G) = \{1, 2, 3, 4, 5, 6\}$$

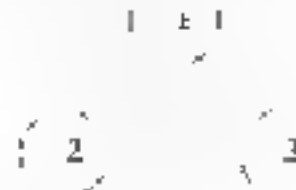
$$E(G) = \{(1, 2), (2, 3), (3, 4), (3, 5), (3, 6), (4, 5), (5, 6), (6, 3)\}$$

حيث V تمثل مجموعة من النقاط و E تمثل مجموعة من الحواف

1.5 أنواع المخططات (Type Of Graphs)

يوجد نوعين من المخططات هي

1 المخطط غير المتجه (Undirected Graph) هو المخطط الذي تكون العلاقة بين عناصره غير متجهة أي لا اتجاه تكون غير مهمة كما في الشكل رقم (1.7)



$$V(G) = \{1, 2, 3\}$$

$$E(G) = \{(1, 2), (2, 3), (3, 1)\}$$

شكل رقم (1.7) مخطط غير متجه

2- المخطط المتجه (Directed Graph).

هو المخطط الذي تكون الحافة بين عنصرين مرتبة، بمعنى أي من الإحداثيات تكون هوية، ومسبوقة كما في الشكل رقم (18).



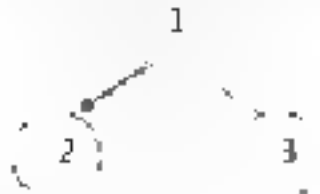
$$V(G) = \{1, 2, 3, 4\}$$

$$E(G) = \{(1, 2), (1, 3), (3, 4), (4, 3)\}$$

شكل رقم (18) مخطط متجه

2- المخطط المشترك (Undirected Graph & directed Graph)

هو المخطط الذي يحتوي على النوعين كما في الشكل رقم (19) التالي -



$$V(G) = \{1, 2, 3\}$$

$$E(G) = \{(2, 1), (1, 3), (2, 3)\}$$

شكل رقم (19) مخطط مشترك

العنصر هو مجموعة من المتغيرات التي تربط بين متغيرين في المخطط

مثلاً: في الشكل رقم (20) التالي يوجد عنصر بين نقطتين لهاتين (1, 5) و (3, 5).



شكل رقم (20) بوصف مخطط يجرى بهجوم عد مسارات

الحل :

1. نعتبر الإنعاش للنقطة (1,5) هي (1,3), (3,5) وليس (3,6), (2,3), (1,2).
2. نعتبر الإنعاش للنقطة (1,6) هي (1,3), (3,6) وليس (3,5), (2,3), (1,2).

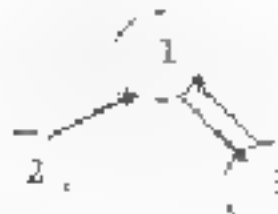
مع ملاحظة أنه لا يمكن كتابة المسار باستخدام أي من المجموعتين { }

3.4 طول المسار (Path Length) :

هي عدد المستقيمت (الخطوط التي تربط نقطتين في المخطط كما نلاحظ في مخطط الشكل رقم 21)

- أ. طول المسار من (1) إلى (2) ويمكن أن يتخذ طولين آخرين، هو (3) إلى (6,1) طر. المسار بينهما هو (3) ويمكن أن يتخذ الطول الآخر وهو (2)

والمعرفة طول المسار فإنه عند أن نصل النقاط من نصل عدد 4 وراجع (عدد المستقيمت). في المخطط المسج أحيف يوجد أكثر من مسار بين نقطتين وبالتالي فإنه نعيد مشكلة تكرر في طول المسار (مسار متغير) كما في الشكل رقم (21) التالي.



1. (1,3), (3,4)
2. (1,3), (3,1), (1,3), (3,4)

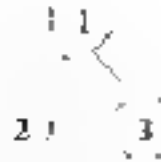
شكل رقم (21) بوصف مسارات مستخدمين الاتجاه

المخطط المتصل (Connected Graph)

هو المخطط الذي يوجد فيه مسار بين أية نقطتين من نقاط المخطط

المخطط غير المتصل (Dis-Connected Graph)

هو المخطط الذي تكون بعض نقاطه غير متصلة ببعضها البعض، بمعنى وجود الشظ (22) التالي يوضح ذلك.



مخطط متصل غير متجه



مخطط متصل متجه



مخطط متصل غير متجه



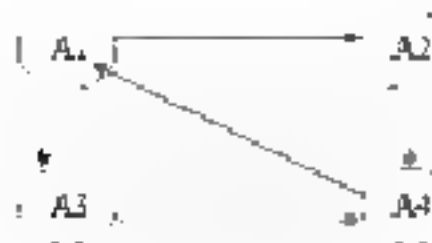
مخطط غير متصل متجه

شكل رقم (23) يوضح أنواع المخططات

يمكننا استخدام المخططات لتحديد أو معرفة أقصر المسارات من خلال إيجاد كل المسارات

وهي ثم بيان الإجمالي فيها

على أن يكون المخطط التالي



// المطلوب //

1. توضيح دور α المحطط
2. تفتي المحطط بمصفوفة
3. بيان قيم المصفوفة
4. بيان تأثير الدخلة والخارجة من كل نقطة
5. إيجاد أقصر مسار بين نقطة A_1 ونقطة A_4

// الحل //

1. المحطط متجه وتمثل $\langle \text{Direct Graph \& Connected Graph} \rangle$
2. يمكن تغيير المحطط بالمصفوفة بالتالي

	1	2	3	4
1	A_{11}	A_{12}	A_{13}	A_{14}
2	A_{21}	A_{22}	A_{23}	A_{24}
3	A_{31}	A_{32}	A_{33}	A_{34}
4	A_{41}	A_{42}	A_{43}	A_{44}

3. يمكن بيان قيم المصفوفة بأعداد كالتالي

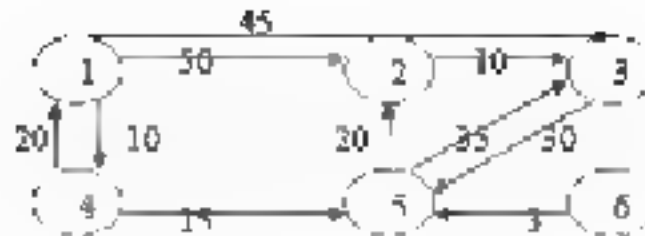
		A_1	A_2	A_3	A_4
2	A_1	0	1	1	0
1	A_2	0	0	0	1
1	A_3	0	0	0	1
1	A_4	1	0	0	0
		1	1	1	2

4. إيجاد مجموع α تقييم الدخلة والخارجة من كل نقطة قال
مجموع التقييم في كل صف يمثل عدد الخطوط الخارجة من كل نقطة
(Row = Out degree)
مجموع α التقييم في كل عمود يمثل عدد الخطوط الدخلة للنقطة
(Column = Input degree)

العدد النقطي	المسارات الداخلية	المسارات الخارجية
A_1	2	1
A_2	1	1
A_3	1	1
A_4	2	1

5. إيجاد أقصر مسار بين نقطة A_1 ونقطة A_4
 a المسار (A_1, A_2, A_4) b المسار (A_1, A_3, A_4)
وهو أقصر المسارات لأن درجتهما تساوي (2)

مثال 11 شبكة المخطط التالي



المطلوب:

- 1- حدد نوع المخطط
- 2- حدد المصفوفة
- 3- حدد قيم المصفوفة
- 4- حدد العيود الداخلة والخارجة
- 5- حدد المسارات بين نقطته
- 6- أعط القيم المسار هاتين النقطتين (1,5) و (1,6) و (2,6)

الحل:

- 1 للمخطط متصل وبعينه
- 2 المصفوفة هي

	1	2	3	4	5	6
1	A11	A12	A13	A14	A15	A16
2	A21	A22	A23	A24	A25	A26
3	A31	A32	A33	A34	A35	A36
4	A41	A42	A43	A44	A45	A46
5	A51	A52	A53	A54	A55	A56
6	A61	A62	A63	A64	A65	A66

- 3- حدد قيم المصفوفة كالآتي

	A1	A2	A3	A4	A5	A6
A1	0	50	45	10	0	0
A2	0	0	10	0	0	0
A3	0	0	0	0	20	0
A4	20	0	0	0	15	0
A5	0	20	30	0	0	0
A6	0	0	0	0	3	0

4. تحديد المسارات الداخلة والخارجة

المسار كـ الداخلة	المسار كـ الخارجة	سم النقطة
20	105	A1
70	10	A2
50	30	A3
10	35	A4
49	55	A5
0	2	A6

5. إيجاد أقصر مسار بين النقطتين

- المسار بين (1,6) هو " لا يوجد مسار بينهما
- المسار بين (1,5) هو (4,5) ، (1,4)
- المسار بين (2,6) هو " لا يمكن التوصل إلى النقطة (6) لعدم وجود مسار داخل إليها

4. طريقة الجشع (Greedy Method)

في هذه الطريقة نستخدم غالباً لحل مسائل الأمثلية (Optimization problems) التي علمه
 هي تكون (Maximum) ثلثي، معين أو تصغر (Minimum) غير الأمكن الثمن الشيء ، كـ
 في حلالة الربح أو الضارة

في هذه المسائل نحاول على العنصر التالية-

- دالة هدف (Objective Function) وهي تمثل دالة هدف في نطاق معين. فكل حل ممكن
 ويكون حل ممكن، وسعى أفضل الحلول الممكنة وهو الأفضل .
- في مجموعة القيود (Constraints) جوان مجموعة الحلول التي يمكن اختيار
 ممكن حلول ممكنة (Feasible Solutions) في حل ممكن الذي يعطي دالة هدف يساوي
 الحل الأمثل (Optimal Solution)

في الطريقة هذه الطريقة يمكن أن

يتم الحل الأمثل في طريقة الجشع على مراحل ففي كل مرحلة يُحدد أفضل قرار فبعض
 أمثلة هذه الخطوات في التبرير أمثلة من بعض الخطوات يجب أن نحقق أمثلة في

ملاحظة: إن أغلب المسائل التي نطوّر طريقة الجشع تكون من أكثر من واحد إلى الفصل
 (n input) حيث نهدف إلى إيجاد الحل الذي يحقق هو الأمثل ولكن قد يكون هو الأمثل
 لا فهذا يعني أنه يوجد مسألتان أو مشاكل في هذه الطريقة .

يوجد نموذجين لطريقة الجشع هما

1. نموذج المجموعات الجزئية (Subset Paradigm)

في هذا النموذج نبحث عن مجموعة جزئية من المجموعة الأصلية بحيث تكون هي الأمثل ويجب
 أن نحقق هذه المجموعة قيود المسألة ، وفيه يلي تجريد مبسط أو تحكمي لهذا النموذج .

```

SolType Greedy (type a[],int n)
a[1..n] contains the n inputs.
{solType solution=Empty, initialize the solution
For(int i=1; i<=n; i++)
{ typeX= Select(a);
  If feasible(solution,X)
    Solution= Union(solution,X);
}
return solution;

```

إن المقصود من (Solution) هي مجموعة العناصر التي لا تحتوي أي عنصر في المجموعة ذاتها (Subset) فهي حالة تحتل بها مجموعة مختارة ونرجع واحد من العناصر لم يتم الاختيار هل إلى أصل هو حل ممكن؟ نعم فإنه يصلح هذا الحل لمجموعة الحلول السابقة وإلى حالة يمكن فإنه يندرج في الحد الأدنى

4.5 مسألة حقيبة (Knapsack Problem)

سنأخذ مسألة الجراب أو حقيبة الظهر كمثال نموذج للمجموعة التجريبية (Knapsack Problem) حيث هناك مجموعة كلف بوصف أي هذه الخفية

مجموعة العناصر (N) من الكيفيات حيث يمكن أن تكون أي شيء وأنت جراب مسطحة (C) مقاساً بالكيلو غرام (لاحظ كم مختلف وزن من هذه الكيفيات)

لتكن (I) المجموعة (W) حيث (1 ≤ i ≤ n)

بصفة الكمية (x_i) وهو يمثل حل لمسألة حيث (0 ≤ x_i ≤ 1) يحقق ذاته هي (P(x))

مختار - في الجراب بحيث نعلم القيمة المحسنة أي أنه هناك مسأله (Maximum)

دالة الهدف هي معيار الأمثلية (نفسه الكمية بعد الدالة هي معيار الأمثلية أي الحل الأمثل)

$$\text{Maximize } \sum_{i=1}^n P_i x_i$$

ثم طباعة النتيجة (X) حيث بالاعتماد على قيمة النتيجة فإنه يمكن جلاء من الخلفه صنف أو لا صنف

في نموذج المسألة يجب أن نحقق

$$\sum_{i=1}^n W_i x_i \leq C$$

1 أي أي حل يحتوي ظهر المسألة هو حل ممكن

2 أي حل يحقق كثير بعد زيادة الهدف هو الحل الأمثل

والنتيجة هي

$$0 \leq x_i \leq 1 \quad 1 \leq i \leq n$$

and

$$P_i \neq 0 \quad \forall i \neq 0 \quad 1 \leq i \leq n$$

ومتلخص الآن مثالاً لتطبيق هذه الطريقة .

$$P[1, 3] = \{25, 24, 15\}, C = 20, n = 3$$

$$W[1, 3] = \{13, 15, 0\} \quad \begin{matrix} P_1 \\ P_2 \end{matrix} \quad \begin{matrix} 1 & 3 & 1 & 6 & 1 & 5 \end{matrix}$$

في هذه الطريقة تعطى ثلاث جدول عمقهم مختلف باختلاف العمق كما يأتي:
الجدول الأول : أنه أول حل محتمره يكون هو الحل الأمثل بالقيمة
الحل الثاني : أنه أول حل محتمره يكون هو الحل الأمثل بالوزن
الحل الثالث : أنه أول حل محتمره يكون بالاعتماد على سعة تقسم الخفض على الواحد المقطرة .
سوف نقوم بعد جدول لتوضيح حلول المعادلة

المعامل	$\sum P_i W_i$	$\sum P_i C_i$	$x_1 \quad x_2 \quad x_3$
المعادلة الأولى بوزن	20	28 2	$(1, 2/15, 0)$
الوزن الأقل أولاً	20	31	$(0, 2/3, 1)$
المعادلة الأمثل لكل وحدة وزن أي $(\frac{P_i}{P_j})$ بوزن	20	31 5	$(0, 1, 1/2)$

وبالتوضيح هذه النتائج بتطبيق التعبير الأول كالتالي:

$$20 \cdot 8 = 2$$

$$2 \cdot 15 = 2/15$$

$$0 \cdot 10 = 0/10 = 0$$

في المعادلة (3) تعبر عنه وحدة وأساري (1) بحيث بعد طرح الوزن من المعادلة يمكننا
ننظر بهذه المعادلة حتى يتم على الحروف
مع ملاحظة النتائج للخبر : أعلاه بالمحظوظ المعيار الثالث هو الذي يعطى بالنتيجة الغير قائمة
ومثالاً لن هي $(2, 2, 3)$ المعادلة له تعال الحل الأمثل.

ولذلك من صحة النتائج يمكننا إجراء عملية ضرب بين قيم المعيار النتيجة والوزن في
الخاصة بها .

وهذا هو الإجراء المتبع في تنفيذ خوارزمية Knapsack Problem

```

Void Greedy Knapsack (float m, int n)
// p[1..n] and w[1..n] contains profits and weights
// respectively of the n-objects ordered such
that  $p_i/w_i \geq p_{i+1}/w_{i+1}$ .
// m is the Knapsack size and x[1..n] is the solution vector
    n هو قسمة عدد الكائنات (m) على سعة الحقيبة ، ويجمع الكائنات بالترتيب
    من الأعلى إلى الأسفل ، حيث يصعب الترتيب تنازلي أثناء الحل وبالتالي فإنه من
    الممكن الانتقاء للحل الأول والذي يعال لأنه هو في النهاية هو المقبول X يكون فرج
    مهماً أي يعبري كيف صغيره
{
    For(int i=1; i<=n; i++)    x[i]=0.0;
    float U=m;
    For(j=1; j<=n; j++)
    {
        If (W[j] > U) break
        X[j]=1.0;
        U = U - W[j];
    }
    If (j<=n)    x[j]=U/W[j];
}

```

مميزات خوارزمية

تتطلب هذه الخوارزمية بعض الخطوات من وقت ترتيب الكائنات ابتدائياً $O(n)$ من الوقت فقط حيث إننا نحقق ترتيباً بوقت يكون صفياً ووقت الترتيب حيث أنه في حالة استخدام خوارزمية ترتيب بالوقت الترتيب هو $n \log n$ لأنها تكون دائماً وتعطي حلاً أمثل

ملاحظة: يوجد خوارزمية أخرى (Knapsack 0/1) حيث أنها هي نفس النتيجة أو لا تحلها وبالتالي فإنه يجب حلها بطريقة (If) الأخيرة من هذه الخوارزمية وفي هذه الحالة لا يكون هناك صفراً للحصود على حل أمثل

2. نموذج الترتيب (Ordering paradigm).

في هذا النموذج يتم اتخاذ القرارات باعتبار المسلمات بتوقيت معين بحيث كل قرار يتم اتخاذه باستخدام معيار التفضيل والذي يمكن حسابه من خلال تفرعات المسألة السابقة

ملاحظه ، في مثال الترخيص هناك قبل واحد يحصل بمرور في تغير الحل

لترتيب هذا النموذج نورد بعد مسألة بمعنى مسألة الحفاظ الترخيص الأمثل **Optimal Merge Pattern** حيث تتلخص هذه المسألة بالمعنى التالي

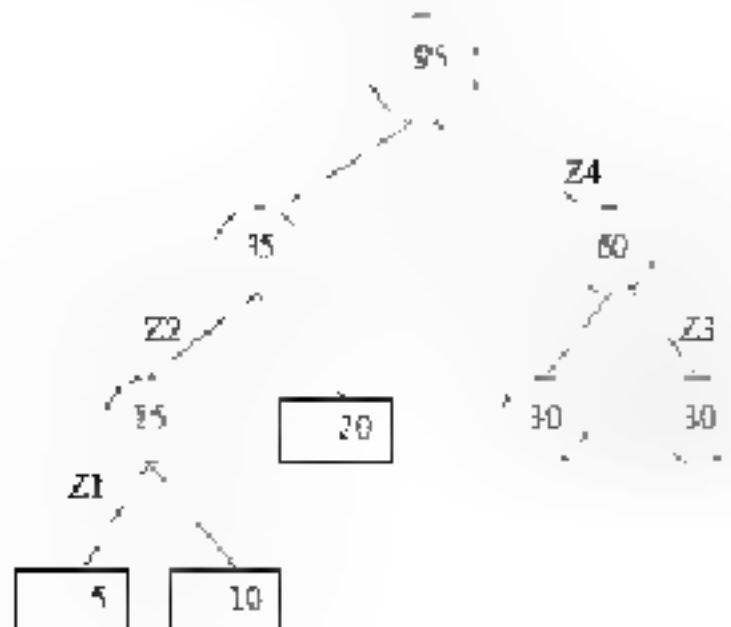
يوجد أربع مجموعة ملفات متخزنه بحيث يكون لدينا ملف واحد لكل وجنا 5 ملفات وكل تحريك بتقريب الختمة بملفات المدمجة أي انه ستكون مسالة (M-Number) .
 من هذه المسألة هي مسألة دمج (D) من الملفات الموزعه على شكل ابروج وبتالي ملف واحد مرتب بكل عدد ممكن من الحركات المسلمات ، من هذه المسألة تستدعي ترتيب هذا من ابروج من الملفات المراد دمجها لتلك فهي تتعلق بنموذج الترتيب

من ناحية المخطط في

(التلخيص حركة السجلات المدمج بتقريب 4000 حجمه 50 كل خطوة أولاً).

مثال / يجب معرفة الملفات المرصده كما في التالي.

5 في (20 30 10.5,50) [F1 F5]



إن القيمة (20) تمثل عدد القنود في الشجرات في الصف الأول، وهكذا بالتسوية يتبعه 30 في القمة (30) تمثل عدد القنود في الصف الثاني.
 في القيمة (10) تمثل طول الصف
 في ترتيب التفرع نجد الترتيب هو الآخر:

20,30,LS 30
 30 30 35
 60,35

إذا كانت W تمثل عدد الصفوف الخارجيه للصف W في W يمثل طول الصف W
 في الجدول التالي نحرك الصفات لشجرة التفرع التالية هذه تكون

$$\sum_{i=1}^n W_i^2$$

$$= 5^2 + 10^2 + 20^2 + 30^2 + 30^2$$

$$= 205$$

وهذا هو الحد الأدنى الذي يكون هو الحجم الأقل للصف
 نطبق تطبيق عملية تفرع عشوائي:

وهذه هي الدالة التي تولد الشجرة للتطبيق مكتوبة بلغة C++:

```
Struct Tree node
{
    Struct tree* Lchild,*Rchild;
    Int Weight;
}
```

إن الجزء علامه حاصه بشكل العدة للشجرة الشافية بالقيمة الموصولة

```
Typedef Struct Tree node type;
```

```
Type *tree (int n)
```

List is global list of n single node binary trees as described above.

n تمثل عدد الصفات W في صف الصف للقيمة W تمثل عدد القنود لكل صف

0	W	0	n
---	---	---	-------	---

```
For (int i=1, n=n, i++)
```

```
{ type *pt=new type;
```

```
Pt->Lchild=Least (list);
```

```
Pt->Rchild=Least (list);
```

```
Pt->Weight=(Pt->Lchild->Weight)+(Pt->Rchild->Weight);
```

```
Insert (list,*pt);
```

```
}
```

إن هذا ال for يكون خاصه بتعبيره التفرع بين الصفات ، الدالة Least ترجع مؤشر إلى
 النقطة التي تكون ذات أقل وزن من الصفات ويحفظ مكانها ، الدالة insert تقوم بإضافة
 عنصر إلى القائمة list الخاصة بالعدد W ، *pt هو رقم محطرات المؤشر ،

```
Return (Least(list));
```

```
}
```

إن الناتج من هذه العملية هو مؤشر إلى شجرة التفرع التالية

و قمنا بالي توضيح موجد الخوارزمية

1. كل شجرة في القائمة List ستكاد تصبط عدة راحة ، هذه الراحة هي عدة خارجة حيث تتكرر من ثلث جهات في (Rchild) ، (Wright) ، (Lchild).
(Rchild) و (Lchild) متوالت هما متقربة ، إذ تمثل (Wright) محتوى على طرف بعد تعلقت لهم بعضي
2. الدالة (Tree) ستعمل تلتس بحريني لها (Least) و (Insert) حيث إن الدالة (Least) تقوم بخلق شجرة في القائمة حتى إذا كانت لا تزال لا يوجد الدالة مؤش إلى هذه الشجرة و نقوم بخلق الشجرة نألي حفرها Pt إلى الدالة (Least) حيث إن مؤش هذه نقطة هي Pt

3. في شجرة التمدج لتتلقه لتتلقه في جهة هذه الخوارزمية تستخدم لتحديد انه سلك سم بعضها حيث يتجر التمدج على تلك السلك التي ستكاد لتتلق الإكبر في الشجرة

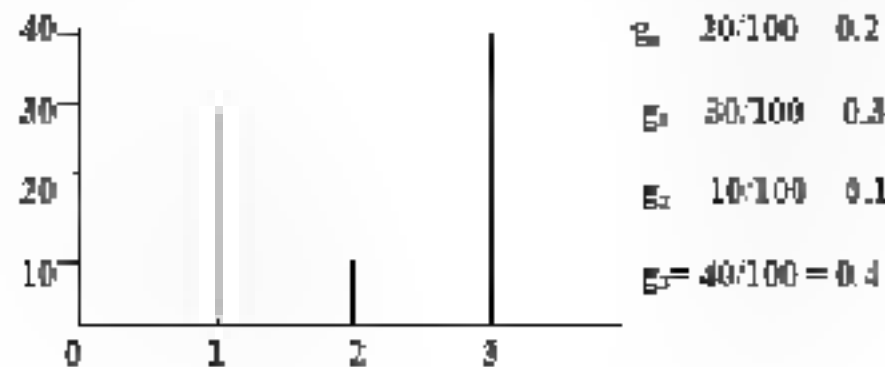
تعددت الوقت تخيرونه

إن جهة for الرئيسية تتكرر (n-1) من المرة ، في حلة الاحتفاظ بالقائمة (List) من جهة تعدد بدأ سعة بي هم الزور في الحقول من الدالة (Least) تتلق (O(1)) من الوقت والدالة (Insert) تفعل لخلقها بوقت هو (O(n)) أي إن الوقت الكلي المستغرق هو (O(n²)).

الحل - مستخدم قائمة التمدج من بجد لتتلق البيانات -

طريقة هوفمان (Huffman code) سبقت بمدة للعالم هوفمان 1952 ، تعتمد على فكرة تقليل البيانات وصنعها بكون التغير على عدة المتغيرات يبتور فقدان البيانات

هوفمان / ليف ليفان طريقة هوفمان لتتلق المجموع Greedy mile مسألة بالمرحج التكراري التتلي



مثال ضم المبرج



ب. ترتيب القيم وجميع القيمتين



ج. الاستمرار بجمع بقية الأصغر 5 أصغر الرسوم بقيمتين فقط

مقود الا عتيايه Original gray level (natural code)	الاحتمالية Probability	معدرة هرفمان Huffman code
$g_0 : 00_2$	0.2	010_2
$g_1 : 01_2$	0.3	00_2
$g_2 : 10_2$	0.2	011_2
$g_3 : 11_2$	0.4	1_2

يجد Entropy الخاص بالاحتمالية المستخدمة

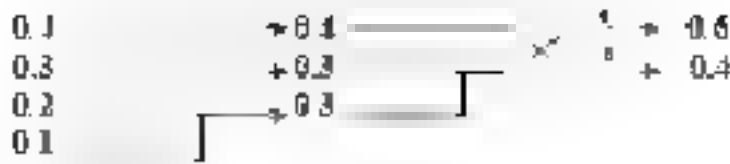
$$\begin{aligned} \text{Entropy} &= -\sum_{i=0}^3 P_i \log_2 (P_i) \\ &= [(0.2) \log_2 (0.2) + (0.3) \log_2 (0.3) + (0.1) \log_2 (0.1) + \\ &\quad (0.4) \log_2 (0.4)] = 1.916 \text{ bits / pixel} \end{aligned}$$

كما ان يوجد $\log_2 (X)$ يمكن الحصول عليه حسب القانون التالي:
 $\log_2 (X) = 1.322 * \log_{10} (X)$

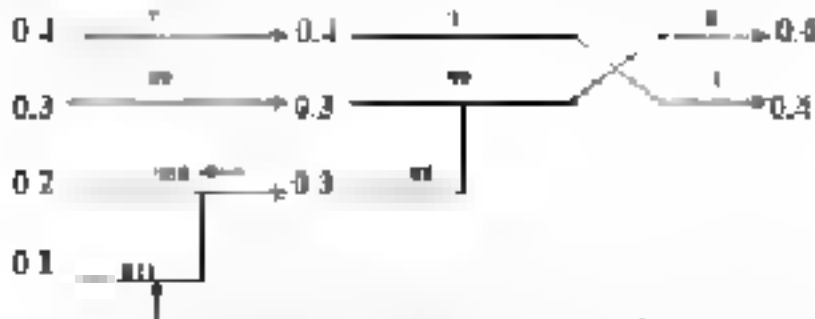
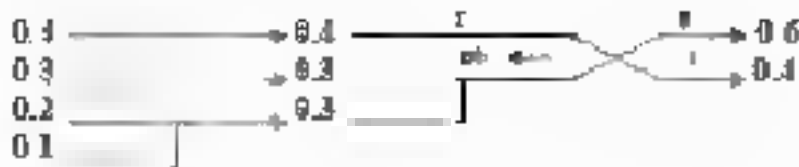
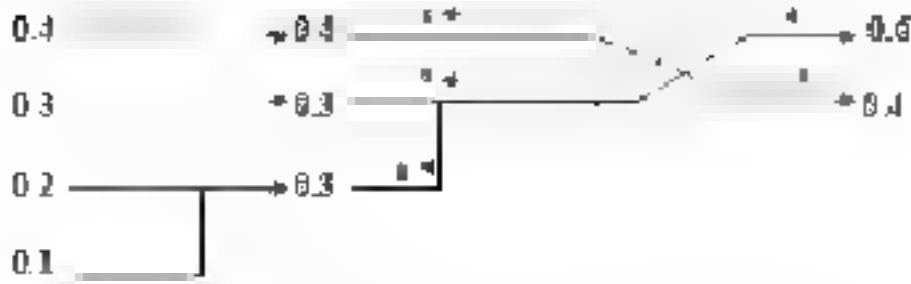
يجد معدل طول الشفرة حسب القانون التالي (Average length)

$$\begin{aligned} \text{Lave} &= -\sum_{i=0}^3 L_i P_i \\ &= 1(0.2) + 2(0.3) + 3(0.1) + 1(0.4) = 1.9 \text{ bits / pixel} \end{aligned}$$

مطور التفرع



مطور غير حجب



مثال: لنفترض أننا نستخدم استراتيجية واحدة للدمج، وتحديد سرعة الفرع

Horizontal code Example (from our test)

Commit	Author	File	Path
1	John	main	main
2	John	main	main
3	John	main	main
4	John	main	main
5	John	main	main
6	John	main	main
7	John	main	main
8	John	main	main
9	John	main	main
10	John	main	main
11	John	main	main
12	John	main	main
13	John	main	main
14	John	main	main
15	John	main	main
16	John	main	main
17	John	main	main
18	John	main	main
19	John	main	main
20	John	main	main
21	John	main	main
22	John	main	main
23	John	main	main
24	John	main	main
25	John	main	main
26	John	main	main
27	John	main	main
28	John	main	main
29	John	main	main
30	John	main	main
31	John	main	main
32	John	main	main
33	John	main	main
34	John	main	main
35	John	main	main
36	John	main	main
37	John	main	main
38	John	main	main
39	John	main	main
40	John	main	main
41	John	main	main
42	John	main	main
43	John	main	main
44	John	main	main
45	John	main	main
46	John	main	main
47	John	main	main
48	John	main	main
49	John	main	main
50	John	main	main
51	John	main	main
52	John	main	main
53	John	main	main
54	John	main	main
55	John	main	main
56	John	main	main
57	John	main	main
58	John	main	main
59	John	main	main
60	John	main	main
61	John	main	main
62	John	main	main
63	John	main	main
64	John	main	main
65	John	main	main
66	John	main	main
67	John	main	main
68	John	main	main
69	John	main	main
70	John	main	main
71	John	main	main
72	John	main	main
73	John	main	main
74	John	main	main
75	John	main	main
76	John	main	main
77	John	main	main
78	John	main	main
79	John	main	main
80	John	main	main
81	John	main	main
82	John	main	main
83	John	main	main
84	John	main	main
85	John	main	main
86	John	main	main
87	John	main	main
88	John	main	main
89	John	main	main
90	John	main	main
91	John	main	main
92	John	main	main
93	John	main	main
94	John	main	main
95	John	main	main
96	John	main	main
97	John	main	main
98	John	main	main
99	John	main	main
100	John	main	main

Original symbols			Step 1 reduction			
Symbol	P_i	Order	q	r	s	t
a_1	0.4	1	0.4	0.1	0.4	0.1
a_2	0.3	2	0.3	0.2	0.3	0.1
a_3	0.1	3	0.1	0.3	0.1	0.5
a_4	0.1	4	0.1	0.4	0.1	0.4
a_5	0.1	5	0.1	0.5	0.1	0.3
a_6	0.1	6	0.1	0.6	0.1	0.2

Although it might not look like a truly both uniquely decodable and instantaneous code. Think about how you'd decode an incoming bit stream.
Entropy of the example

$$H = -(0.4 \log(0.4) + 0.3 \log(0.3) + 0.1 \log(0.1) + 0.1 \log(0.1) + 0.1 \log(0.1) + 0.1 \log(0.1))$$

$$H = 2.44$$

Average code length

$$E = 4(1) + 3(2) + 1(3) + 1(4) + 1(5) + 1(6)$$

$$E = 1.8$$

Now that $H < E$, as expected, there is no code that is optimal (lossless) to find any code which did better (i.e. closer to the optimal entropy H). That is why the Huffman code is a 'compact' code.

شجرة هوفمان مستخدمة لتشفير النصوص

الطريقة المستخدمة لتشفير كلاً من

- 1- نص، جعله نصية
- 2- حساب تكرار كل حرف في النص
- 3- عمل شجرة «نلك» بحيث لكل نصين في كل فرع
- 4- ترقيم الشجرة بحيث كل مسار على اتجاه اليسار يعطى 0 وكل مسار على اتجاه اليمين يعطى 1.
- 5- كتابة شفرة حرفي لكل حرف في النص، خلال سبع خطوات
- 6- إيجاد معدل الترميز و Entropy

مثال، لووجد شفرة هوفمان للجملة التالية

dead beat safe decided dad. Dad faced a faced-ab Dad a
 needed, dad be back.

Space	a	b	c	d	e	f	.
1"	12	4	5	10	12	4	4

2 جمع اول هفتین یکل عرب

Space	a	b	c	d	e	f	.
7	12	4	5	19	12	4	4

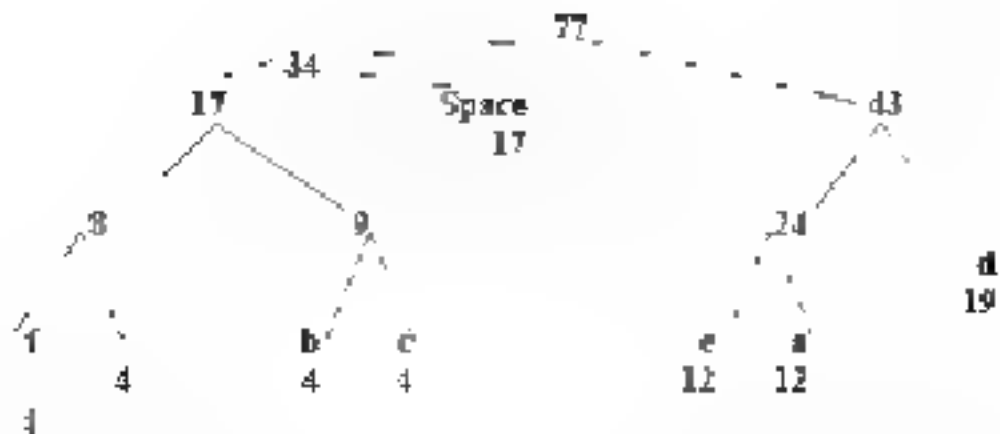
Space	a	b	c	d	e	f	.
17	12	4	5	19	12	4	4

Space	a	e	b	c	d	f	.
17	12	12	4	5	19	4	4

8	17	34	17	43	12
12	24	12			

8	17	34	77	43	19
12	24	12			

3. يرمز الشجرة



Space	a	b	c	d	e	f	.	رمز
01	101	0010	0011	11	101	0000	0101	شجرة هو

وحيث أن الخوارزمية الخاصة ببناء شجرة هو هافمان هي:

Huffman(C)

1. $n = |C|$
2. $Q = C$
3. for $i = 1$ to $(n - 1)$ do
4. allocate a new node z
5. $z \text{ left} = x = Q \text{ Extract-Min}$
6. $z \text{ right} = y = Q \text{ Extract-Min}$
7. $f[z] = f[x] + f[y]$
8. Insert(Q, z)
9. ...return $Q \text{ Extract-Min}$ return the root of the tree

حيث يمكن تقليل التعيد من $n \log(n)$ إلى n^2

الفصل الخامس
البرمجة الديناميكية
**Dynamic
programming**

Complexity	10	20	30	40
n	0.00001 sec	0.00002 sec	0.00003 sec	0.00004 sec
n^2	0.0001 sec	0.0004 sec	0.0009 sec	0.0016 sec
n^3	0.1 sec	0.8 sec	2.7 sec	6.4 sec
n^4	1 sec	16 sec	243 sec	256 sec
n^5	101 sec	320 sec	2430 sec	10240 sec
2^n	0.02 sec	10 min	20 years	256000000 sec

مثال: نوضح حركية إيجاد أقصر مسافة

What is $d[i, j]^0$?

$$d[i, j]^0 = \begin{cases} 0 & \text{if } i = j \\ \infty & \text{if } i \neq j \end{cases}$$

What if we know $d[i, j]^{m-1}$ for all i, j ?

$$\begin{aligned} d[i, j]^m &= \min(d[i, j]^{m-1}, \min_k (d[i, k]^{m-1} + w[k, j])) \\ &= \min(d[i, k]^{m-1} + w[k, j]) \quad 1 \leq k \leq n \end{aligned}$$

since $w[k, k] = 0$

This gives us a recurrence with which we can evaluate d in a bottom up fashion

```

for i = 1 to n
  for j = 1 to n
     $d[i, j]^m = \infty$ 
    for k = 1 to n
       $d[i, j]^0 = \text{Min}(d[i, k]^m, d[i, k]^{m-1} + d[k, j])$ 

```

This is a $O(n^3)$ algorithm just like that x shortest path problem but it only goes from m to $m+1$ edges

5- التكرار الأول: المسافات بين النقاط والمراكز
في هذه الخطوة يتم حساب المسافة بين كل نقطة ومركز التجمع الجديد، كما في الخطوة
التالية، ينتج لدينا مصفوفة من المسافات

6 التكرار الأول: جميع النقاط
على مركز الخطوة التالية: نحيل كل نقطة إلى مركز تجمع بالإعتماد على أقل مسافة بالمجموعة
في مصفوفة المسافات المحددة بنقل النوى التالي (B) إلى المجموعة الأولى، بعدها نتلى باقي
النوية كما هي تظهر مصفوفة المجموعات

7 التكرار التالي: تحديد مركز التجمع
الآن نمرر بمجموعة الخطوة الرابعة بحساب تحديد مركز التجمع الجديد بالإعتماد على
عنايه التجميع في التكرار الأول حيث تكون كل من المجموعة الأولى والثانية من عناصره

8 التكرار التالي: المسافات بين النقاط والمراكز
تكرر الخطوة التالية، ينتج لدينا مصفوفة مسافات جديدة

9 التكرار التالي: قصص النقاط
نمرر بنوى نحيل كل نقطة إلى مركز تجمع بالإعتماد على أقل مسافة

نتبع لدينا في النهاية معادلة التجميع بين التكرار الأول والتكرار الثاني، لاحظ في
المجموعة يتم تغير من حيث عناصرها وهذا يعني أن عملية الحساب في الـ (k-mean
clustering) وصلت إلى حالة الثبات، وهذا يعني أن هذه الخوارزمية لم تعد بحاجة إلى تكرار
في التكرار، وبالتالي حصلنا على النتيجة النهائية للتجميع

2 خوارزمية 2 (Subtractive Clustering)

المشكلة في طريقة التجميع السابقة (Mountain Clustering) هي أن المصنف الحسابي
يبدأ بطريقة مبدئية أعلى المشكلة، وذلك لأنه كما نرى سابقاً قدم تقسيم الـ Mountain
Function عند كل نقطة تقاطع في الشبكة على محورتي الـ x و y
استخدمت خوارزمية الـ (Subtractive Clustering) حل هذه المشكلة وذلك عن طريق عدد
من نقاط البيانات لتكون مركز المجموعة، بدلاً من استخدام نقاط بعض خطوط الشبكة كما هو
الحال في الـ (h/c)، وهذا يعني أن المعادلة الحسابية أصبحت تتكيف مع حجم المشكلة بدلاً من
المعادلة

خوارزمية الـ (Subtractive clustering) هي عملية تحديد مركز المجموعة على أن
تجميع صفه مسرعة بين كل الـ عناصر دون قطع بعد المجموعة المقترحة لتبدأ
وبعد هذه الطريقة على حساب كثافة البيانات عند كل نقطة ضمن مستوى معين، لهذا كثافة
كل نقطة مرشحة لتكون مركز تجمع، فكله يمكن إيجز كثافة البيانات عند نقطة x من المعادلة
التالية

حيث أن π ثابت يجب أن يكون ثابتاً حول كل نقطة يتم حساب الكثافة عنده هذه المعادلة
وكما نرى هذا القطر أصبح لدينا عدد اثنين من المجموعات، وكله في القطر را، عدد
المجموعات، وبأن تكون قيمة ρ أكبر من قيمة π غالباً يستخدم $\rho = 1$ ، وذلك لتقليل
تجوز الكثافة عند نقاط المجاورة لنقطة المركز الأولى

تم حساب الفرق الأول $sc1$ والذي كان كثافة الحواف عند أعلى ما يمكن $\{D_1\}$ بعد ذلك يتم حساب قيم الكثافة بجدية بعد كل خطوة \dots

وتقوم خوارزمية الـ *Subtractive clustering* بالخطوات التالية

1. إيجاد نقطة معينة موجودة في المجال تكون عند أعلى كثافة عالية ويتم حساب الكثافة من المعادلة الأولى وهو يتم اختيار نقطة معينة كمرکز ، وبذلك عن طريق وجودها بين عند كثير من النقاط المحيطة .

2. يتم حذف نقاط النجم

3. يتم تبحث الخوارزمية عن مركز جديد وذلك عن طريق حساب قيم الكثافة للنقاط الأخرى كما في المعادلة الثانية ، وتستقر هذه العملية حتى لا يتضاءل من كل النقاط أو يوجد عدد كافٍ (مما يثبت) من المعجم عند

بعد إجراء هيرش هذه الخوارزمية هي لها بكثر فعالية من الخوارزمية التي تكونت سابقاً كما أنها الأسرع في أشكال المعطيات

4-5 خوارزمية (Dijkstra)

يخمسج ديكسترا (Edsger Dijkstra) هو أحد العلماء الهولنديين في علوم الحاسب ، ولد فيمسج الهولندي الأصل سنة 1910م في مدينة روتردام ، ويد مشواره التعليمي بهجلاً فيزيائية نظريته في جامعة ليبير ، كان من علم ما أتت له أن أضافته منصب في علوم الحاسب

استلم ديكسترا عام 1972م جائزة *J.M. Turing* على نظريته في الحاسب في برمجة اللغة كما حظت بمنصبه في (كرسي سكرتيرجورنسون) نظريته في الحاسب) في جامعة تكساس في بوسطن منذ عام 1984م وحتى تقاعد عام 2000م

من أبرز إسهاماته في علوم الحاسب هي خوارزمية نظرية لأقصر المسارات والتي تعد أيضاً بحوارزمية سيكسترا ، سميت هذه الخوارزمية في تنظيم نقل المعلومات بين أجهزة الحاسب وعرفها بعد بحوارزمية الطريق لأقصر المسار

كتب ديكسترا عام 1958م ورقة بحثية هامة في تخصصات نظم البرمجة الحاسب وعملات النماذج السيمولية وانتشر بحثاً بديكسترا ، عبارة البرمجة بشيبرد (تسمى في أكثر تستخدم تلك بـ "I more use a for 2") والتي تعبر إلى حقيقة أنه حين تجد نفسك تقدم أكثر من عمل فحبه معلوم عليه لوجه حتى يوجد سلك من هذا المسار داخل حله بكناريه

والخوارزمية للحصه بيده الطريقة هي

$DIKSTRA(G, w, s)$

1. INITIALIZE_SINGLE-SOURCE(G, s)
2. $S \leftarrow \{ \}$ S will ultimately contain vertices of final shortest-path weights from s
3. Initialize priority queue Q i.e. $Q \leftarrow V[G]$
4. while priority queue Q is not empty do
5. $u \leftarrow EXTRACT_MIN(Q)$ \therefore Pull out new vertex

```

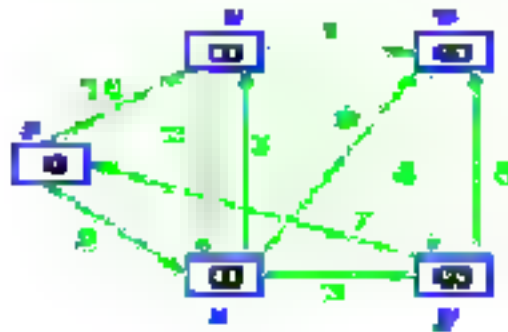
6   S ← S ∪ {u}
    Perform relaxation for each vertex v
    adjacent to u
7   for each vertex v in Adj[u], do
8   Relax (u, v, w)

```

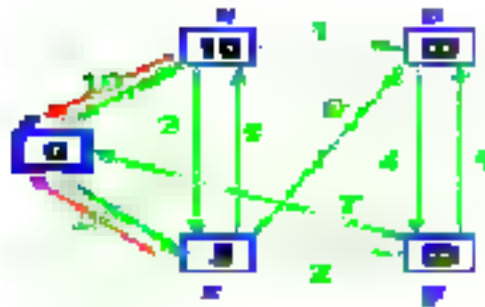
5.5 إنشاء تطبيق خوارزمية (Dijkstra)

مثال // تطبيق الخوارزمية كما في الخطوات التالية

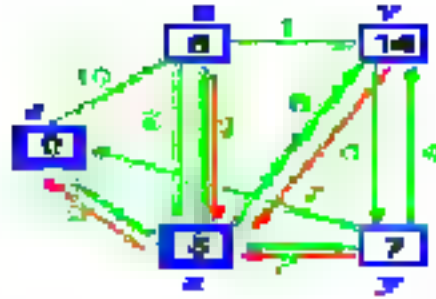
1 الاسم مستطى $G=(V, E)$ كل قبة هناك كات عبر سوية عا العدة S حيث كات Q



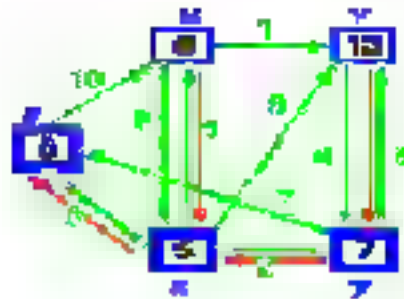
2 أولا نحدد عقدة أوله من S ونحدد $d[S]$



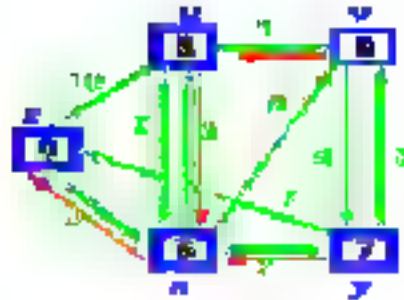
3 الأخير عدة x ونطبق خطوات الخوارزمية



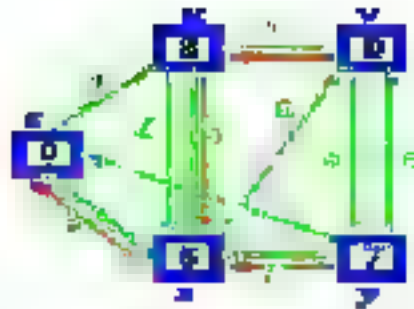
4. اكتب ψ قيمة البعد ψ وبتطبيق هذه الخطوات



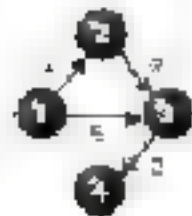
5. الآن هناك المقام ψ نحرك عقدة ψ جاريها V



6. عقدة ψ المسقط سوف يعطى أصغر مسار ψ بغير تصريف العقدة



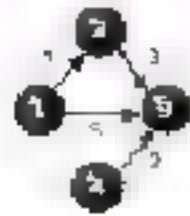
مثال ١ : نقوم بتطبيق الطريقة المقترحة على المسألة الآتية



$$D(0, j)$$

	1	2	3	4
1	+	+	5	+
2	+	+	+	+
3	+	+	+	+
4	+	+	+	+

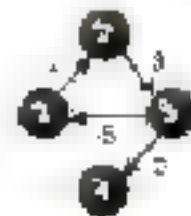
a



$$D(0, j)$$

	1	2	3	4
1	+	+	5	+
2	+	+	+	+
3	+	+	+	+
4	+	+	+	+

b



$$D(0, j)$$

	1	2	3	4
1	+	+	5	+
2	+	+	+	+
3	+	+	+	+
4	+	+	+	+

c

let $C = \{1, 2, \dots, n\}$ denote the set of cities and for each city j in C let $P(j)$ denote the set of its immediate predecessors and let $S(j)$ denote the set of its immediate successors, namely set

$$P(j) = \{k \in C : D(k, j) < \text{infinity}\}, j \in C$$

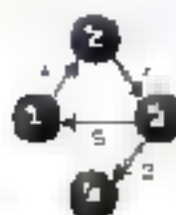
$$S(j) = \{k \in C : D(j, k) < \text{infinity}\}, j \in C$$

Thus, for the problem depicted in Figure 1 a) $P(3) = \{0\}$, $P(2) = \{0\}$, $P(4) = \{0, 1\}$, $P(5) = \{0, 1, 2\}$, $S(0) = \{1, 2, 3\}$, $S(1) = \{4, 5\}$, $S(2) = \{5\}$, $S(3) = \{6\}$, $S(4) = \{6\}$, $S(5) = \{6\}$ where $\{\}$ denotes the empty set

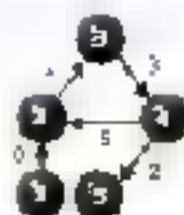
Answer: If multiple methods are used, there is immediate notification and/or if there is no notification, there may be immediate surveillance testing.

$$NP = \{ \langle n, E, R \rangle, n = \{1\} \}$$

$$N5 = \{ \langle n, 5 \rangle \mid n \in \mathbb{N} \} = \{1\}$$

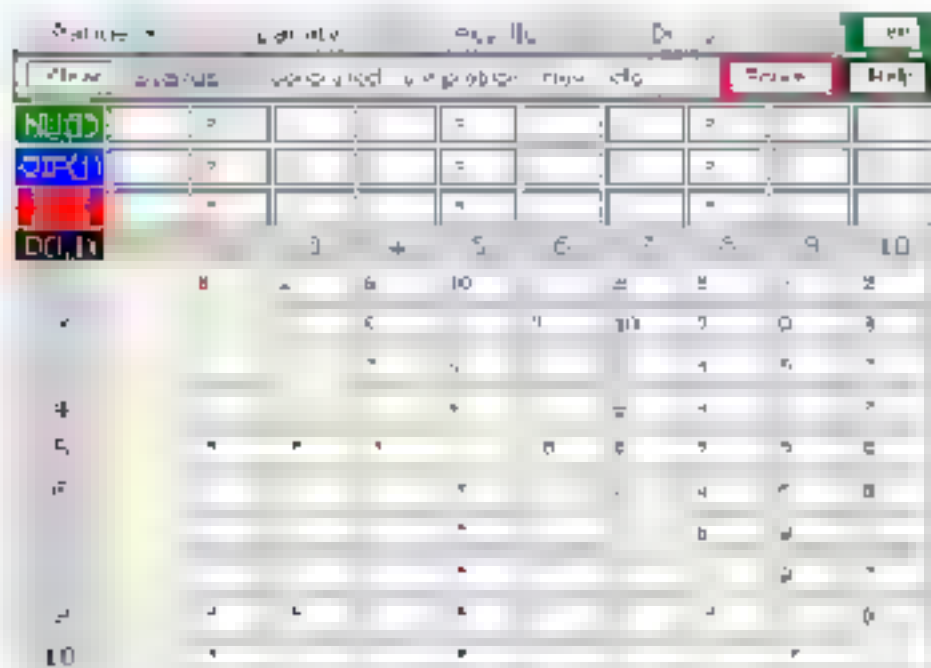


100,000	1	2	3	4
1	70	1	10	70
2	70	10	3	70
3	-5	10	10	2
4	70	10	10	70



	1	2	3	4	5
1	0	1	2	3	4
2	1	0	3	4	5
3	2	3	0	5	4
4	3	4	5	0	1
5	4	5	4	1	0

上



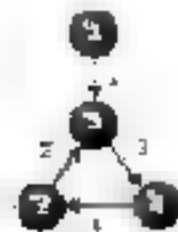
يتم هذا إيجاد الصور المتساوية

x_{ij} Quantity (Flow) sent along the link from city i to city j
 $i=1, \dots, n$

then the minimum cost network flow problem is as follows

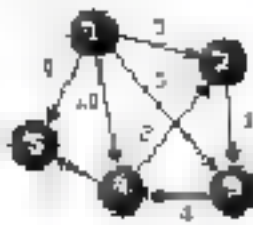
$$\begin{aligned} \min \sum_{i,j} f_{ij} x_{ij} \\ \text{s.t.} \quad \sum_{j=1}^n x_{ij} - \sum_{k=1}^n x_{ki} = a_i \quad i=1, \dots, n \end{aligned}$$

مثال: توضيح الطريقة



$$\begin{aligned} f_{12} &= 0 \\ f_{13} &= 4 + f_{43} \\ f_{24} &= \min \{2 + f_{42}, 1 + f_{14}\} \\ f_{34} &= 3 + f_{43} \end{aligned}$$

مشاركون



f_{ij}	2	3	4	5
1	4	3	10	9
2	+	+	5	+
3	+	2	+	+
4	+	2	+	1
5	+	+	+	+

a

f_{ij}	2	3	4	5
1	4	3	10	9
2	+	+	5	+
3	+	2	+	+
4	+	2	+	1
5	+	+	+	+

b

الص

Iteration k	J	F	Iteration k	U	F
0	$\{1,2,3,4,5\}$	$(0, \infty, \infty, \infty)$	0	$\{1,2,3,4,5\}$	$(\infty, \infty, \infty, \infty)$
1	$\{2,3,4,5\}$	$(1, 5, \infty, \infty)$	1	$\{2,3,4,5\}$	$(1, 5, \infty, \infty)$
2	$\{3,4,5\}$	$(2, 5, 2, \infty)$	2	$\{3,4,5\}$	$(1, 5, 2, \infty)$
3	$\{4,5\}$	$(0, 5, 3, 7, 9)$	3	$\{4,5\}$	$(0, 5, 3, 7, 9)$
4	$\{5\}$	$(0, 5, 3, 2, 8)$			
	a			b	

$$F_{k+1} = F(5) = 9 > F(a) = F(b) = 7.$$

الخطوة التالية للعثق تكون كالآتي

Initialize $k=1$ $F(1) = 0$ $F(\infty) = \text{Infinity}$ $j = \infty$ n
 $U = \{1, 2, 3, \dots, n\}$
 Iterate While $(U) \neq \emptyset$ and $F(j) < \text{Infinity}$ Do
 $u = \min(U)$
 $F = \min_{j \in U} \{F(j), D(i, j) + F(i)\}$ $j = u$
 $U = \arg \min \{F \text{ in } U\}$
 End Do

Range	Sparsity	Acyclic	$D(i,j) \leq 0$	Gen					
File	Status	Generated new problem	New algo	Solve Help					
N=10	?	?	?	?	?	?	?	?	?
DIP(i,j)	?	?	?	?	?	?	?	?	?
D(i,j)	?	?	?	?	?	?	?	?	?
D(i,j)	?	?	?	?	?	?	?	?	?
			4	5	6	7	8	9	10
1	6		6	10	7	8	5	3	2
2		3	5	7	7	0	7	9	
3			2	6	3	0	4	6	
4				4	7	0	4	7	
5					10	5		3	5
6							4	6	8
7							6	9	
8								9	
9									8
0									

هذه هي الطريقة المستخدمة في حركة الأربوب ويجب أن تكون متساوية

Example arena:

```

X-----
B   B
B-----
B-----
BB-----
--B-----
      B.B
--B-B-----
B----- Y

```

Input

```

0 0
. 0
. 1
. 6
2 0
3 2
4 4
4 5
5 4
6 5
6 7
7 0

```

Sample Output:

```

0 0
0 1
0 2
. 2
2 2
2 3
2 4
2 5
2 6
3 6
4 6
5 6
6 6
7 6
7 7
. 1

```

PROGRAM (TRIED AND TESTED IN TURBO C++

```
#include <stdio.h>
#include <string.h>
struct Matrix { short int **array,
                int row,int col;
                };
struct Vertex { int num; int curDist
                };
typedef struct Matrix matrix;
typedef struct Vertex vertex;
void getGrid(matrix &m)
void getShortestPath() /* Dijkstra Algorithm */
void printSolution,int p[],int index);
matrix m;
int main()
{
    getGrid(m)
    getShortestPath()
    printf("n 1 1'm a");
    free(m.array);
    return 0;
}
void getGrid(matrix &m)
{int ctr1,ctr2 blockedSquares,x,y
scanf("%d%d%d",&m.row,&m.col,&blockedSquares)
m.array=(short int **)malloc(m.row*sizeof(short int *)),
for(ctr1=0;ctr1<m.row;ctr1++)
m.array[ctr1]=(short int *)malloc(m.col*sizeof(short int));
for(ctr1=0;ctr1<m.row;ctr1++)
for(ctr2=0;ctr2<m.col;ctr2++)
m.array[ctr1][ctr2]=0;
for(ctr2=0;ctr2<blockedSquares;ctr2++)
{
    scanf("%d%d",&x,&y);
    m.array[x][y]= 1
}
}
```

```

void getShortestPath() /* Uses Dijkstra Algorithm */
{
    int ctrl = 0, ctr2 = 0, row1, col1, row2, col2,
    int *predecessor = (int *)malloc(sizeof(int) * mrow * mcol);
    vertex *toBeChecked, minVertex,
    toBeChecked = (vertex *)malloc(sizeof(vertex) * (mrow * mcol + 1));
    for (ctr1 = 1; ctrl <= mrow * mcol; ctrl++)
        predecessor[ctrl] = 3.000;
        toBeChecked[ctrl].num = ctrl - 1,
        toBeChecked[ctrl].currDist = 3.000;

    predecessor[0] = 0;
    toBeChecked[0].num = toBeChecked[0].currDist = mrow * mcol;
    toBeChecked[1].currDist = 0;
    while (toBeChecked[0].num != 0)
    {
        minVertex = toBeChecked[1], ctr2 = 1;
        for (ctr1 = 1; ctr1 <= toBeChecked[0].num; ctr1++)

            if (toBeChecked[ctr1].currDist < minVertex.currDist)
                {ctr2 = ctr1; minVertex = toBeChecked[ctr1]}
            },
        toBeChecked[ctr2] = toBeChecked[toBeChecked[0].num];
        toBeChecked[0].num =
        row1 = minVertex.num / mcol; col1 = minVertex.num % mcol;
        for (ctr1 = 1; ctrl <= toBeChecked[0].num; ctr1++)

            row2 = toBeChecked[ctr1].num / mcol;
            col2 = toBeChecked[ctr1].num % mcol;
            if (marray[row2][col2] == 0)
            if (((col1 < col2) * (col1 < col2) == 1 && row1 == row2) || ((row1
                row2) * (row1 - row2) == 1 && col1 == col2))
            if (toBeChecked[ctr1].currDist > minVertex.currDist + 1)

                toBeChecked[ctr1].currDist = minVertex.currDist + 1,
                if (toBeChecked[ctr1].num == 0)
                predecessor[toBeChecked[ctr1].num] = minVertex.num;
            }
        },
    }
}

```



```

printSolution(predecessor m,row*m.col-1);
}
void printSolution(int p[],int index)
{
    if(index==0)
        printf("0 0");
        return;
    };
    f(p[index]==31000)
        return;
    printSolution(p,p[index]);
    printf("%d %d",index/m.col,index%m.col);
}

```

6-5 المسحط متعدد المراحل (Multistage graph)

هو مسطح موجبة قيمة تقسم بعدد إلى $(K \geq 2)$ المجموعات منفصلة (V_1) حيث $(1 \leq i \leq K)$ المجموعة (V_i) تتكون من بحيث $(|V_i| \geq 1)$ (أي أنه عند العناصر الموجودة $i=1$) حيث V_1 المجموعة الأولى والثانية تحتوي على عقد واحد

أقر من S إلى T في عدة اتجاهية في V_1 وإلى T في عدة اتجاهية في V_K وأقر من V_i إلى V_{i+1} في عدة اتجاهية بين $(i, i+1)$ كما في المخطط التالي

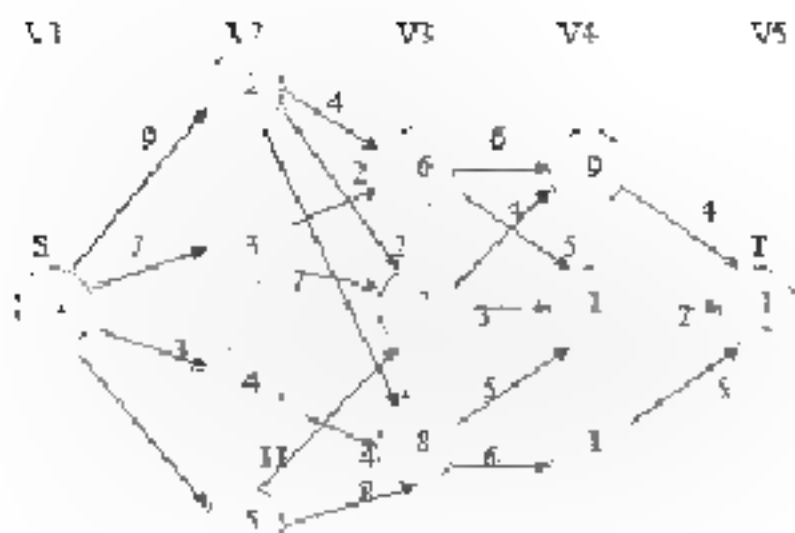
$$C_{i+1}(r, s) = \min_{t \in V_{i+1}} \{C_i(r, t) + C_{i+1}(t, s)\}$$

كف ير من الحالة الموجودة بكل مسطح $(i, i+1)$

إلى كلفة المسار من البداية S إلى النهاية T في مجموع كلف الحواف على المسار

ملاحظة 1: مسلة المسطح متعدد المراحل في إيجاد مسار أقل كلفة من $(S$ إلى $T)$

كل مجموعة V_i تمثل مرحلة في المسطح وكل مسار من $(S$ إلى $T)$ يبدأ (بمرحلة 1) وينتهي (بمرحلة K).



شكل رقم (24) يوضح مخطط التدفق العرسل

صياغة البرمجة التليميكية بحسب مخطط I_k من مراحل

6-3. الطريقة التصاعدي (Forward approach)

- (i) نلاحظ من كل مسار من S إلى T يكون نتيجة تعاقب $(k-2)$ قرار ، حيث k قرار (i) لنفرض نحدد فيه عقدة $(V_k + 1)$ حيث $(2 \leq k \leq K)$ تكون على العنبر P_{k-1} هو العنبر k كلفة من العقدة (i) في العنبر P_{k-1} إلى العقدة (ii) يعطى إن $P(2,2)$ حيث (i) يعقل عقده (ii) يعقل مرحلته من السراجل والى (iii) يعقل كلفة تلك العنبر يعطى $Cost(2,2)$ وبمستعمل الطريقة التصاعديه (أي أنها بحل تقاسمياً) يعطى ذلك إننا تبدأ من الأخير (العقدة T) إلى الأولى (العقدة S).

$$Cost(i, r) = \min_{r \in J_i + 1 \leq j \leq E} \max \{c(i, j, r) + \cos t(j+1, r)\} \quad 1$$

وهي كلفة المسار الأقل كلفة حيث

في ذلك $\langle r, r \rangle$ هي حكة تنتمي إلى جميع الحواف الموجودة في المخطط بحيث

$$Cost(k+1, j) = \begin{cases} c(j, r) & \text{if } j < j+1 \leq E \\ \infty & \text{if } j < j+1 \leq E \end{cases}$$

ولهذا فإن المعقنة رقم (1) تقطع للحالة $Cost(2,5)$ بصفاً إلى

$$Cost(k+2, r), \forall r \in F_{k+2}$$

تتبع ذلك

$$Cost(k+3, j), \forall j \in F_{k+3}$$

وهكذا نسير على أن نصل في قطع الموجود في $P(1)$ وبغاية $Cost(2,5)$ حيث نجد
أخرى قيمة للمسار الأقل كلفة

ويعبر المخطط الموجود في الشكل رقم (2) السابق بصل على

$$Cost(3,6) = \min\{5 + \cos t(4,9), 5 + \cos t(4,10)\} = 7$$

وهي قيمة المسار الأقل كلفة

$$\cos t(3,7) = \min\{4 + \cos t(4,9), 3 + \cos t(4,10)\} = 7$$

$$Cost(3,8) = \min\{5 + \cos t(4,10), 6 + \cos t(4,11)\} = 7$$

$$Cost(2,2) = \min\{4 + \cos t(3,6), 2 + \cos t(3,7), 1 + \cos t(3,8)\} = 7$$

$$Cost(2,3) = \min\{2 + \cos t(3,6), 7 + \cos t(3,7)\} = 9$$

$$Cost(2,4) = \min\{11 + \cos t(3,8)\} = 18$$

$$Cost(2,5) = \min\{11 + \cos t(3,7), 8 + \cos t(3,8)\} = 15$$

$$Cost(1,1) = \min\{9 + \cos t(2,2), 7 + \cos t(2,3), 3 + \cos t(2,4), 2 + \cos t(2,5)\} = 16$$

لاحظ أن الحد للكلف الأقل بصورة متتالية دائماً أنه الأصغر حيث أن آخر كلفة مسار تم
حسابها هي كلفة $Cost(1,1)$ وتعني كلفة $Cost(S)$

ويبدأ من المسار الأقل كلفة هو S إلى T ، كان بلكه متعارفاً 16 واختياد المسار بسجل
المراتب المستخدمة في كل حالة (عند)

ونفترض أن $D[i]$ هي قيمة T التي تصغر العلاقة التي ذكرناها سابقاً وهي

$$c[i, r] + \text{Cost}(\{i + 1, r\})$$

ملاحظة: // في $D[i, r]$ يمثل القرار المسد للمرحلة القادمة حيث ذكرى بحقه التي تعطي الآن
قيمة حسب العلاقة السابقة

ملاحظة: // في قدم القرار r المسد هي نفس الرقيم للخط الموجوده على المسار ورائف سابقاً
بالمسار (2-2)

$$D[2,2] = 7, D[2,3] = 6$$

$$D[2,4] = 8$$

$$D[2,5] = 8$$

$$D[1,1] = 7$$

وباعتبار المسار الأقل كلفة هو

$$S = 1, V_2, V_1, \dots, \dots, V_5, T = 12$$

حيث

S تمثل النقطة الأولى وهي الأقل كلفة

$$7 = V(2)$$

$$7 = V(3)$$

$$10 = V(4) - 1$$

$12 = T$ وتمثل النقطة الأخيرة

$$V_2 = D[1,1] = 7$$

$$V_1 = D[2, D[1,1]] = D[2,7] = 7$$

$$V_4 = D[2, D[1,1]] = D[3,7] = 10$$

ملاحظة: // يجب تحديد النقطة الأولى بغير إلى ذلك الموال التالي ما هي النقطة الأقل التي يجب

أن نختارها في المرحلة التالية

مستخرج من المسار الأفضل هي $(T) (12, 10, 7, 2, S)$

والآن سوف نقوم بكتابة الخوارزمية لحل هذه المسألة (خوارزمية المخطط متعدد المراحل

المسطرة لتطبيقه الصاعدي)

تتضمن خوارزمية المخطط متعدد المراحل المداخلة الطريقة الصاعدي، في العقد V مرتبة من

1 إلى n من عقد المسألة S يعني الممرس 1 ثم تعطي عقد المجموعة V_2 الممرس متتالية

حتى تصل إلى عقدة النهاية T أي في الممرس المسطرة بعد المجموعة $V_2 + 1$ تكون أكبر من

تلك المسطرة $V[i]$

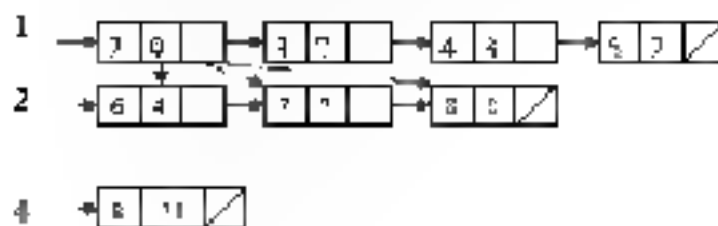

```

Struct node
{
    int vertex;
    float weight;
    Struct node*Link;
}

```

- `vertex` : هو القيمة الموجودة على الحافة التي تمثل بالمعبر `weight` -> ضيف هذا الجزء من البرنامج مع تمثيل واحدة فقط
- `Link` : هو السلسلة المتصلة بعد فتح كائني

```
Struct node*headnodes[n];
```



headnodes



- لاحظ ان التمثيل الأولي تم ربط برؤية فقط يعني انك استخدمنا `headnodes`
- ان عملية انشاء وفتح البيانات ونقش المخطط يجب ان تتم قبل تنفيذ الخوارزمية `FGraph` ومن ثم يتم استخدامها

وقدنا في جزء البرنامج الخاص بهذه الطريقة التصاعدي (Forward approach)

```

FGraph ( Type head*node.)
{
    Type *p=newType;
    p->Link;
    p->weight;
    p->vertex;
    for(int i=2;i<=m;i++)
    {
        type *p
        p->weight;
    }
}

```

```

        p->vertex;
        q->link=p;
    }
    q->link=0;
}
head[u]=0;

void FGraph (Type *head[26],int k,int n)
{ float cost [100],x, int p[100],D[100]
  cost[n]=0;
  for (int j=n-1;j>=1;j--)
  {cost[j]=head[j]->weight+cost[head[j]->vertex],
   D[j]=head[j]->vertex;
   Head[j]=head[j]->link;
   While (head[j] != 0)
   { x=head[j]->weight +cost[head[j]->vertex]
     If (x<cost[j])
     { cost[j]=x,
       D[j]=head[j]->vertex;
     }
     head[j]=head[j]->link;
   }
  }
}

P[1]=1, p[k]=0
for(int j=2;j<=k-1;j++)
  P[j]=D[p[j-1]];
for(int i=1,i<=k-1,i++)
  cout<< p[i]<< " ";
}

```

تعبير بعبارة الدالة (FGraph)

في حالة تنفيذ الخوارزمية القائمة على التكرار (Linked list) فلا يمكن إيجادها في وقت يتناسب مع درجة العقدة (j) (مع ارتباطات العقد مع العقد التي بعدها) (الأسهم) حيث (j) هي من 1 إلى n لذا كل الخوارزمية $O(E)$ أي بحرف قاي وقت بدرجة fox الأولى هو $O(|V| + E)$ حيث V يمثل عدد العقد و E هي عدد الحواف في هذا E هي عدد الحواف و V هي عدد الحواف

هو بخصيصه فوقه حولة f_{opt} النجبه لهر $\Theta(k)$ حيث k يمثل عدد المراحل
هذا يعني ان التكلفة بالكلية للحواري منه هي

$$\Theta(k^2 + k)$$

بالإضافة إلى الخزن، التي تنطبقه امتدادات "توجد حجة بوجود خزن التكلفة في المصنفين
PD

2-6-5: الطريقة التكرارية (Backward approach):

إن تحديد التبع هذا يكون من النهاية إلى البداية أي من (T إلى S) حيث هذه الطريقة حلاً
مباشرياً وعكس اتجاه حساب التكلفة مع كالاتي.

$$b_{\text{cost}}(i, j) = \min_{\substack{r \in E, r \leq j \\ s \in V, s \geq i}} \{ b_{\text{cost}}(i-1, r) + c[r, j] \} \quad \dots \quad 1$$

يتم كل البعد المرحلية في المرحلة التالية $b_{\text{cost}}(2, j)$ في نفس التكلفة البعد التي تربط نقطة
2، في بعده التالية

$$b_{\text{cost}}(i, j) = \begin{cases} \infty & \text{if } i > j \\ \min_{r \in E, r \leq j} \{ b_{\text{cost}}(i-1, r) + c[r, j] \} & \text{if } i \leq j \end{cases}$$

يتم صياغ هذا كالتالي i, j في نفس التكلفة من البعد S إلى العنصر (j) في المصنف b_{cost}
وكيف (j) b_{cost} هي قيمة المثل $b_{\text{cost}}(i, j)$ يوجد التكلفة b_{cost} عند ذلك يمكن
حساب $b_{\text{cost}}(i, j)$ هناك بحسب $b_{\text{cost}}(i, j)$ بقيمة $i-1$ قبل وهكذا بالتسوية ببقية البعد

المخطط التالي المرسوم في الشكل رقم (24) يوضح لتكلفة في أول مرحلة حسابها هي
التكلفة من هـ يعني عندما ملاحظة الإنهاء للتكلفة للبعد

$$b_{\text{cost}}(3,6) = \min \{ b_{\text{cost}}(2,2) + 4, b_{\text{cost}}(2,3) + 2 \} = 9$$

بصورة للسيطرة على تتبع التبع خطأ أسفل البعد التي تعطي القيمة لأحد طرف بعيدا في
حساب أو الحد التكراري
والتالي بنفس الطريقة بالتسوية لبقية المثلث أي

$$\begin{aligned} b_{\text{cost}}(3,7) &= 1, \\ b_{\text{cost}}(3,8) &= 10 \\ b_{\text{cost}}(4,9) &= 15 \\ b_{\text{cost}}(4,10) &= 14 \\ b_{\text{cost}}(4,11) &= 16 \end{aligned}$$

$$D \leq \cos f(5,12) = 16$$

أما بالنسبة لإيجاد أقرب من فلز القصور الأقل كلفة من (T إلى S) كان يكلفه مقدار 16
ولتحديد المسار بسجل القصور العشرة في كل حلقة (تعددة) كما يلي

$$D[3,6] = 3, D[3,7] = 2, D[3,8] = 2$$

$$D[4,9] = 6, D[4,10] = 7, D[4,11] = 8$$

$$D[5,12] = 10$$

$$P[1] = 1$$

$$P[4] = 10$$

$$P[3] = 7$$

$$P[2] = 2$$

$$P[n] = 12$$

وباعتبار المسار الأقل كلفة هو

$$S = 1, V_2, V_3, V_4, T = 12$$

حيث

S يمثل العدد الأولي وفي الإثن دائما

$$2 = V(2)$$

$$7 = V(3)$$

$$10 = V(4)$$

$$12 = T \text{ يمثل العدد الأخير}$$

$$V_2 = D[3, D[4,10]] = D[3,7] = 2$$

$$V_3 = D[4, D[5,12]] = D[4,10] = 7$$

$$V_4 = D[5,12] = 10$$

نستنتج إن المسار الأمثل هو : (T) 12, 10, 7, 2, 1(S)

وهذه هي الخوارزمية الشخصية بالخطوط معتمد على حل المسألة بالطريقة الشخصية
(BGraph)

```

void BGraph (Graph G ,int k ,int n ,int 2[])
{ float cost [maxsize],
  int D[maxsize] ,r;
  bcost[1]= 0.0;
  for (int i=2; i<=n ,i++)
  {   Compute bcost[i].
      Let r be Vertex Suchthat <r,i> is an edge of G and bcost[r]+c[r][i]
      is minimum;
      bcost[i]= bcost[r] + c[r][i] ,
      D[i]= r;
  }
  P[1]=1, p[k]=n;
  For(j: k-1, j<=2 j-)
      P[j]=D[ p[j+1] ]
}

```

و هذا هو جزء البرنامج الخامس بهذه الطريقة المباشرة

```

FGraph (Type head*node )
{ Type *p=newType;
  q=p;
  p->weight;
  p->vertex;
  for(int i=2; i<=m; i++)
  { type *p;
    p->weight;
    p->vertex;
    q->link=p;
  }
  q->link=NULL;
}
head[n]=0;

void BGraph (Type *head[20],int k,int n)
{ float bcost [100],x; int p[100],D[100];
  cost[n]=0;
  for int j=2 j<=n; j++)
  {bcost[j]=head[j] ->weight+bcost[head[j]->vertex];
    D[j]=head[j] ->vertex;
    head[j]=head[j] ->link;
  }
}

```

```

While (head[j] != 0)
{
    x=head[j]->weight + bcost[head[j]->vertex];
    If (x < bcost[j])
        bcost[j]=x;
        D[j]=head[j]->vertex;
    }

    head[j]=head[j]->link;
}

}

P[1]=1; p[1]=n;
for(int i=b; i<=2; i++)
    P[i]=D[p[i-1]];
for(int i=1; i<=k; i++)
    cout<< p[i] << " ";
}

```

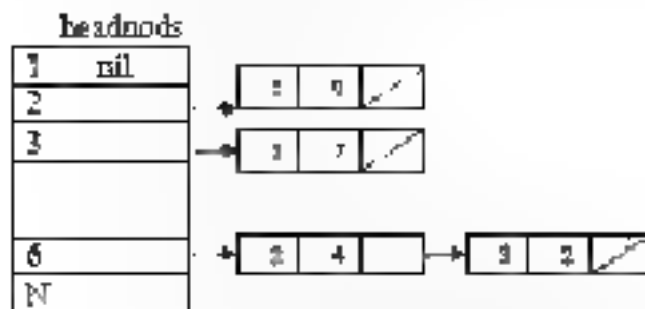
تعيين التبعيات (EGraph)

في تعيينات الوقت والحرر لهذه الخوارزمية هي نفس تلك لتعيينات مخزون رمية (BGraph) ولكن بشرط واحد في تمثيل المخطط ففي الخوارزمية العكسية

هذا يعني في التعيينات الكمية مخزون رمية هي

$$B = |V| + |E|$$

في خوارزمية الحرر، التبعيات تمثل كالآتي:



حيث (V, E) هي مجموعة العقد والحوادث، B هي مجموع العقد والحوادث، P هي مجموعة العقد التي تم تعيينها، D هي مجموعة العقد التي لم يتم تعيينها.

E فصل الجبر ٤٠ الحرف ٧٢ يمثل العدد في قلعة ايسار

ملحوظة: / هناك تعيينات أخرى للمنظمات المعنية للفراد منها ما يخص الوارد (البيع من العمل يقسم على مجموعته فشرائح)

3-6-5 طريقة متتالية الأثر الرجوعا (Back Tracking)

تعتبر إحدى أفكار الطريق للتعلم القصص من حولها، حيث يمكن أكثر من حدث للتعلم (مجموعه من حوال)

الملاحظة / بعد proof نسمح لنا الفهم في تلك اللحظة نبقى كمن

بعض. يعظم المبالغ التي بحيث عن مجموعة حلول أو حل أصلي يحل بعض الحدود
يكون. فكل الصيغة (x_1, x_2, \dots, x_n) حيث x_i هي مجموعة من الحدود هي
هذه. رئيسية لهذه الطريقة هي أنه إذا كانت f هي النتيجة الجبرية $f(x_1, x_2, \dots, x_n)$ من
التي حل أصلي في كل من حلقات التفاضل الجزئية الجزئية هي نفسها

تتطلب العديد من المسائل التي يذو حلها باستخدام طريقة اقتداء الأثر رجوعاً (في تحقيق
الخطى مجموعة من القيود التي يمكن تقسيمها إلى فئتين وهاتى مجموعة من القيود
المجموعة الأولى صريحة (Explicit Constraints) :
وهي قد نعد لتحدد أولاً المسطرة مسطحة هي أن حدث كل للتحقق في منطق القبول
الصحيحة تكون قراءات الحالات الممكنة

المجموعه ثانياً صعيه (Implicit Constraints)

الخطوات المستترة بطريقة التقليل والآثار الجيدة

1. تحديد طرق الحلول وتعظيم العتصمات الإيجابية و الحد من تلك السالبة
2. تقسيم هذا الفرع بطريقة عميقة للكل (محمّد بن مصطفى)
3. بحث الفرع بطريقة شطرنج بوز (Depth First Search) مع استعمال دوال تكبد (Bounding Function) لتجنب الحركات لم يتم غالب حركته لا نقود إلى حق

مثلاً: مسئله صف n ملکه (n queens problem) که ما آن را در فصل ۳، بحث توضیح الگوریتمی روی لوحه شطرنج به صورت $n \times n$ می بینیم، تا آنجا که می توانیم به آن به عنوان مسئله صف n ملکه (n queens problem) می نامیم. این مسئله را می توانیم به صورت زیر تعریف کنیم:

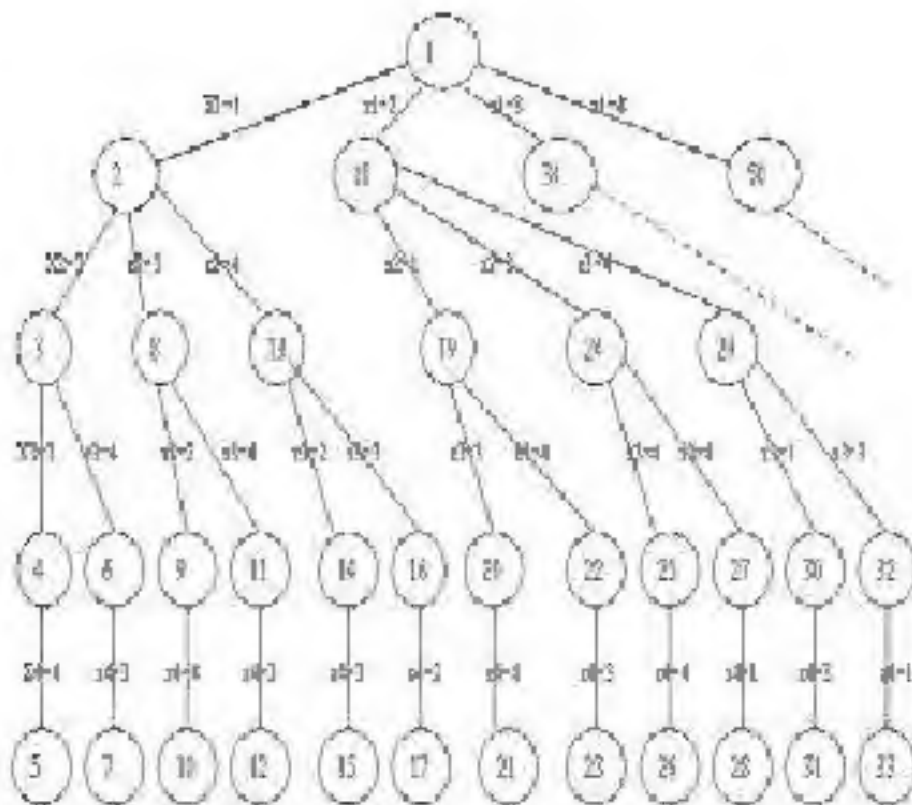
ملاحظة: في هذه الأسفل يوجد نيت لإزالة السمك من الجلد ووضعها على الوجه بطريقة
نظيفة لا يوصى بغيره من ذلك على نفس القصد من الطهي.

لتفرض أن ترتيب صفوف وأعمدة لوحة الشطرنج $[1 \dots n]$ والمكعب أيضا $[1 \dots n]$ يمكن وضع المكعب في الصف i ، عمود j ، بطول عملة m مكعب بالمتجهات $X_1, X_2, X_3, \dots, X_m$ حيث X_i هو العمود الذي توضع عليه المكعب i بمعنى $X_i = 2$ فهذا المكعب i توضع على العمود 2.

القيود الصريحة هنا هي $S_i = [1 \dots n]$

والقيود الضمنية توصف بترابطات X_i بغيرها
القيود الضمنية هي أن كل المكعب يجب أن تكون على أعمدة وأقطر مختلفة.
القيود الصريحة تعطى متجهاتها عندما $n = 3$ ، إن القيد الضمني الأول يشير إلى تشكيل من المتجهات وهذا يفتقر حجم فراغ الحالات أو الحلول من n^m إلى n^3 .

الآن لدينا 4 صور وعلمنا إيجاد للحلول الممكنة ؟
العلماء يعرفون أن الحجم هو 4 لذا يمكن رسم شجرة الاحتمالات (التفصيل) كالتالي :



في هذه العملة يجب أن نراعي الشرط الذي يقول بأن العملة ممكن أن توضع في العمود الأول أو الثاني أو الثالث أي هل نضع كجداية الحل لأن المصفوفة فارغة.

بما إن مجموعة الحلول هي 12 هذا يعني أنه لدينا (24) حالة وإن جزء منها يحقق الحل المطلوب .
 الحراف من قيمة من لكل للقيم الممكنة X_i ،
 الحراف من المستوى i إلى المستوى $i+1$ تحدد قيم X_i بلهذا فإن الشجرة على أقصى اليسار تحتوي على كل الحلول ثلاث X_i تساوي (1).
 إن المسار المتولد من العقد التالية (1,18,29,32,33) يعطي حالة ممكنة يحقق الشرط ، و هو $(2,4,1,3)$ كما في المصفوفة التالية :

	1	2	3	4
1		X1		
2				X2
3	X3			
4			X4	

هذا 12 تقصد بها الملكة الأولى وهكذا بالمتتالية لبقية الملكات .
 أي إن:
 الملكة 1 توضع بالعمود الثاني
 الملكة 2 توضع بالعمود الرابع
 الملكة 3 توضع بالعمود الأول
 الملكة 4 توضع بالعمود الثالث

تمرين // أكتب برنامج يقوم بتطبيق الخوارزمية الخاصة بمسألة (n_queens problem) ؟

References:

- 1-Aho, Alfred V. and Jeffrey D. Ullman [1983]. Data Structures and Algorithms. Addison-Wesley, Reading, Massachusetts.
- 2-Bailer J.:An Introduction to Data Structures ; Allyn and Beacon,Inc,1982.
- 3-Berman A.M.: Data Structures via C++ (objects by Evolution), Oxford University press Inc. ,1997.
- 4-Bertiss A.T. : Data Structures ,theory and practice ; Academic press Inc ,1975.
- 5-Cormen, Thomas H, Charles E. Leiserson and Ronald L. Rivest [1990]. Introduction to Algorithms. McGraw-Hill, New York.
- 6-Dahl O. J. ,Dijkstra E. W and Hoare C.A.R. : Structured 6-programming,Academic press Inc,1972.
- 7-Dale N. and Lilly S.C: pascal plus Data Structures ,Algorithms and Advance programming ,D.C.Heath and company ,1985.
- 8-Goodrich M.T. and Tamassia R. : Data Structures and algorithms in java ;John Wiley and Son Inc. ,1998.
- 9-Gonnet G.H and Beeza -Yates R.:Handbook of Algorithms and data Structures ; Addison_Wesley,1991.
- 10-Horowitz E.and Sahni S.: Fundamentals of data Structures in pascal ;Computer Science press Inc,1987.
- 11-Knuth, Donald E. [1998]. The Art of Computer Programming, Volume 3, Sorting and Searching. Addison-Wesley, Reading, Massachusetts.
- 12-Pearson, Peter K [1990]. Fast Hashing of Variable-Length Text Strings.
- 13-Communications of the ACM, 33(6):677-680, June 1990.
- 14-Pugh, William [1990]. Skip lists: A Probabilistic Alternative To Balanced Trees.
- 15-Communications of the ACM, 33(6):668-676, June 1990.
- 16-Stephens, Rod [1998]. Ready-to-Run Visual Basic Algorithms. John Wiley & Sons, Inc., New York.
- 17-Thomas H. Cormen (Charles E. Leiserson (Ronald L. Rivest and Clifford Stein. Introduction to Algorithms (Second Edition.
- 18-MIT Press and McGraw-Hill, 2001 ISBN7-03293-262-0 .Chapter 1: Foundations, pp.3-122 .
- 19-C.L. PHILIPS& H.T.NAGLE, Digital Control System: Analysis and Design (Prinice- Hall 1984).

- 20-<http://i136.photobucket.com/albums/q175/uraminza/tab1,2,3.jpg>,
- 21-<http://alyaseer.net/files/file.php?id=10>
- 22-<http://akosai.hajznet.com/bubble-sortalgorithm.pdf>
- 23-<http://akosai.hajznet.com/heap-sortalgorithm.pdf>
- 24-<http://akosai.hajznet.com/merge-sortalgorithm.pdf>